Physical Computing als Mittel der wissenschaftlichen Erkenntnisgewinnung

in der Informatik und als fächerverbindende MINT-Arbeitsweise

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik
eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von
Sandra Schulz, M. Ed.

Präsidentin der Humboldt-Universität zu Berlin:
Prof. Dr.-Ing. habil. Dr. Sabine Kunst

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät:
Prof. Dr. Elmar Kulke

Gutachter:

1. Prof. Dr. Niels Pinkwart, Humboldt-Universität zu Berlin
2. Prof. Dr. Maria Knobelsdorf, Universität Wien
3. Prof. Dr. Rüdiger Tiemann, Humboldt-Universität zu Berlin

Tag der mündlichen Prüfung: 26. November 2018

Danksagung

In erster Linie danke ich meinem Betreuer Prof. Dr. Niels Pinkwart, der mir stets mit konstruktivem Feedback, einer positiven Grundeinstellung und viel Vertrauen in meine Arbeit beiseite stand. Des Weiteren möchte ich meinen GutachterInnen Prof. Dr. Maria Knobelsdorf und Prof. Dr. Rüdiger Tiemann für unsere anregenden Gespräche danken. Sie haben mich und diese Arbeit inhaltlich und methodisch sehr bereichert.

Eine besondere Rolle in meiner Promotionsphase spielten Freunde, die mich bereits lange begleiten, mich ständig inspirieren, unterstützen und mich auf den richtigen Pfad zurückbringen, wenn ich ihn verloren habe. Das sind meine Lieben: Markus Nowottnick, Malte Schmidt, Martin Vogt, Wilhelm Tschakert, Danielle Kosian, Natalie Gumprecht, Ruben Assmann und Henriette Wunderlich.

Zwei weitere Menschen sind mir in dieser Zeit sehr ans Herz gewachsen, durch ihr einerseits ständiges Bestreben meinen Karriereweg zu unterstützen und andererseits die Erinnerung, das Büro am Abend zu verlassen. Dr. Márta Gutsche hat mich sehr für diverse Themen sensibilisiert. Nora Butter begleitete mich durch viele menschliche und emotionale Hochs und Tiefs und half mir Stärke als Wissenschaftlerin zu erlangen, wenn die Welt unterzugehen schien.

Meiner Familie danke ich ebenfalls für die ständige Unterstützung und Sicherheit. Sie haben mir stets ermöglicht meine Wege selbstständig auszusuchen und ich musste nur den Mut aufbringen sie zu gehen.

Ich bin sehr dankbar, dass ich an dem strukturierten Promotionsprogramm ProMINTion teilnehmen durfte. Dabei wurde es mir u. a. ermöglicht an der Carnegie Mellon University in Pittsburgh zu forschen. Inspiriert durch dieses Umfeld und die herzliche Unterstützung von Prof. Dr. Bruce McLaren entwickelte ich mein eigenes Forschungsprofil. Auch die ideelle und finanzielle Förderung durch die Deutsche Telekom Stiftung ist hier besonders hervorzuheben.

Einen großen Dank verdienen alle Freunde, die sich bereit erklärten diese Arbeit (z. T. mehr als einmal) gegenzulesen: Christiane Frede, Jennifer Pfennig, Dr. Sven Strickroth, Dr. Lucas Heimberg, Dr. André Frochaux, Benjamin Hauskeller und Eva Sandig.

Nicht zu vernachlässigen sind meine StudienteilnehmerInnen, HelferInnen bei der Transkription und alle, die meine Studien unterstützten. Alle Studien waren sehr aufregend und wären nicht möglich gewesen ohne die Hilfe von: Tim Repke, Martin Vogt und Benjamin Schlotter.

Kurzzusammenfassung

Physical-Computing-Geräten wie Robotern und Mikrocontrollern wird eine wichtige Rolle als Lernmedium für Schülerinnen und Schüler zugesprochen. Zu lernende Kontexte sind ähnlich vielfältig wie die inzwischen existierenden Geräte. Die Komplexität der Systeme ist mannigfaltig und bisherige Forschung geht zumeist von dem Gerät als Forschungsgegenstand aus. Im Rahmen dieser Dissertation wird von einem geräteunabhängigen Physical-Computing-Prozess als Problemlöseprozess ausgegangen, um ein Fundament für nachhaltige und geräteunabhängige Forschung zu schaffen sowie Physical Computing als Unterrichtsgegenstand zu beschreiben. Aufgrund von Merkmalen, wie der Arbeit mit Sensorik und Aktuatorik sowie dem iterativen Testen und Evaluieren, scheint Physical Computing Ähnlichkeiten zu dem naturwissenschaftlichen Experiment aufzuweisen. Dieser Zusammenhang und die potentiellen Auswirkungen auf die Informatikdidaktik werden in den folgenden drei Ausprägungsformen untersucht.

Basierend auf Modellen aus der Literatur wird ein Modell des Physical-Computing-Prozesses abgeleitet und mithilfe empirischer Studien adaptiert. Bei dem Vergleich der Prozesse der wissenschaftlichen Erkenntnisgewinnung und des Physical Computing können diverse Gemeinsamkeiten festgestellt werden. Insbesondere verlaufen die Prozesse parallel zueinander, was die Grundlage für einen MINT-Problemlöseprozess bildet.

Bislang wurden konkrete Probleme von Schülerinnen und Schülern bei der Interaktion mit den Geräten peripher beschrieben. In dieser Arbeit wird eine Analyse von Problemursachen vorgenommen und auftretende Probleme werden kategorisiert. Probleme, die gleichzeitig mehrere Problemursachen haben, werden aufgedeckt und eine Problemtaxonomie zur Beschreibung von Problemursachen abgeleitet.

Ein mehrstufiges Feedback-Modell zur Unterstützung des Problemlösens in Physical-Computing-Aktivitäten wird basierend auf der Problemtaxonomie entwickelt. Durch eine empirische Untersuchung wird es als unterstützend für den Physical-Computing-Prozess evaluiert und bildet damit ein Modell zur Entwicklung von kognitiven Tutorensystemen für Physical Computing.

Abstract

Physical computing devices like robots and microcontrollers play an important role as learning devices for students. These devices as well as the learning contexts are multifaceted. The complexities of the systems are diverse and the existing research is usually concentrated on the devices. This thesis develops as a starting point a device-independent physical computing process by seeing it as problem-solving process. The goal is to construct a base for sustained and device-independent physical computing research and to describe physical computing as a school subject. The physical computing process seems to share similarities with the scientific inquiry process, because of characteristics like working with sensors and actuators and iterativ testing and evaluating. This relation and the implications on computer science education are explored in the following three facets.

Based on existing literature, a model of the physical computing process is derived and supplemented by empirical data. In the comparison of the scientific inquiry and the physical computing processes substantial commonalities are identified. Hence, a base for a joint STEM problem-solving process is built.

So far, concrete students' problems during the activities with physical computing devices are described as a side product. In this thesis problem sources are uncovered and occurring problems categorized. Problems having more than one problem source are uncovered and a problem taxonomy is derived from that.

Based on the problem taxonomy, a multilevel feedback model to support problem solving during physical computing activities is developed. With an empirical exploration, the taxonomy is evaluated. Results indicate that the taxonomy is supportive for achieving the physical computing process. Finally a model for a cognitive tutoring system for physical computing is outlined.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

BPK-Test Basis-Programmierkompetenz-Test

CL Collaborative Learning

CPS Collaborative-Problem-Solving

DBR Design-Based-Research

FF Forschungsfrage

IL Inquiry Learning

ITS Intelligente Tutorensysteme

M Median

MW Mittelwert

NoS Nature-of-Science

PhC Physical Computing

PS Problem Solving

SI Scientific Inquiry

SuS Schülerinnen und Schüler

WoZ Wizard-of-Oz

Kapitel 1

Einleitung

In den letzten 30 Jahren zeichnet sich die Integration verschiedener digitaler Lernmedien in den Unterricht ab (Katterfeldt et al., 2015). Eine Teildisziplin sind Informatiksysteme, die der Robotik zuzuordnen sind, wie z. B. die weltweit verbreiteten LEGO Mindstorms-Roboter. Seitdem Mikrocontroller und Mini-Computing wie von Arduino[1] und Raspberry Pi[2] erschwinglich wurden, werden sie in vielen Anwendungsbereichen eingesetzt. Der Markt ist inzwischen geprägt von einer Gerätevielfalt und Anwendungsbeispielen für alle Zielgruppen. Um diese Fülle an Geräten unter einem Begriff zusammenzufassen, entwickelten sich verschiedene Termini. Zur Beschreibung des Feldes in der Schule und in der Wissenschaft etablierte sich im deutschsprachigen Raum der Begriff *Physical Computing* (vgl. Bildungsserver Berlin-Brandenburg, 2015).
Inzwischen wird Physical Computing (PhC) weiter gefasst, hat somit einen großen Einfluss auf das heutige Leben und ist allgegenwärtig. Dies zeichnet sich zum Beispiel in Form des sogenannten *Maker Movement* ab, welches durch Veranstaltungen wie die *Maker Faire* (Maker Media, Inc., 2017) öffentlichkeitswirksam wird und durch *Maker Spaces* (UnternehmerTUM MakerSpace GmbH, 2016) und *Fab Labs* (Fab Lab Berlin, 2016) (zum Beispiel in Berlin) ständig vertreten ist. In diesem Rahmen wird eine Plattform geschaffen, auf der kreative Ideen ausgetauscht und umgesetzt werden können sowie benötigte Materialien (wie 3D-Drucker, Laser Cutter, Lötkolben) zur Verfügung stehen können. Außerdem werden Workshops zur Anregung der Kreativität und zum Umgang mit den Geräten vor Ort angeboten. Ein Beispielprojekt mit einem Arduino-Mikrocontroller ist in Abbildung 1.1 illustriert. Darin wird eine LED-Ampel in Abhängigkeit zur Umgebungshelligkeit angesteuert. Ein ähnliches Projekt kann in Form von E-Textilien durch einen Lilypad-Mikrocontroller[3] implementiert werden (vgl. Abbildung 1.2). Bislang wird in der Forschung von dem Physical-Computing-Gerät als Gegenstand ausgegangen, ohne dass eine systematische Analyse der Gerätekategorien stattfindet. Nur selten werden didaktische Schlüsse über einzelne Geräte hinweg gezogen. Der Problemlöseprozess und spezifische

[1] https://www.arduino.cc
[2] https://www.raspberrypi.org
[3] https://www.arduino.cc/en/Main/ArduinoBoardLilyPad/

Probleme bei der Arbeit mit Physical-Computing-Geräten werden teilweise sekundär beschrieben. So kann z. B. das Design von Schaltkreisen sowie das handwerkliche Geschick während Physical-Computing-Aktivitäten zu Problemen führen (Kafai et al., 2014b).

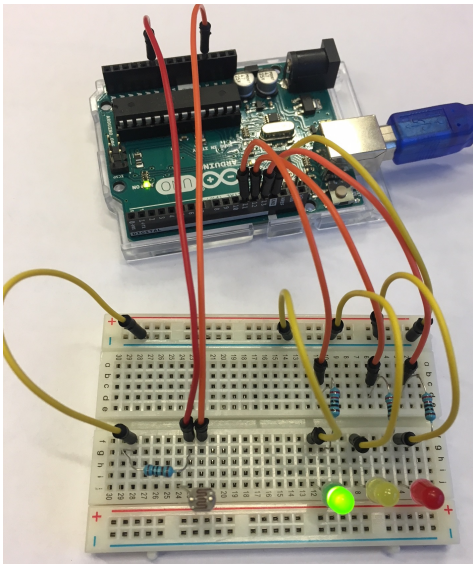


Abbildung 1.1: Beispiel eines Arduino-Mikrocontrollers als Helligkeitsampel

Abbildung 1.2: Beispiel eines Lilypad-Mikrocontrollers als Helligkeitsampel

Durch die Arbeit mit Sensoren und Aktuatoren scheinen im Physical-Computing-Prozess Messunsicherheiten und verwandte Probleme zu entstehen. Diese werden durch Hardwarekomponenten verursacht und sind damit Bestandteil des Problemlöseprozesses. Ähnliche Probleme und Prozessbestandteile (wie das Planen, Testen und Evaluieren) sind aus dem naturwissenschaftlichen Experimentieren bekannt und werden dem Forschungsgebiet der wissenschaftlichen Erkenntnisgewinnung (auch Scientific Inquiry) zugeordnet. Der Prozess der wissenschaftlichen Erkenntnisgewinnung wird in den Naturwissenschaften bereits seit Jahrzehnten untersucht und in verschiedenen internationalen Bildungsstandards als Kompetenz explizit gefordert (Kultusministerkonferenz, 2005; Rocard et al., 2007). Er ist ein wichtiges internationales Bildungsziel zur Entwicklung von naturwissenschaftlichem Verständnis. Da eine Vielzahl von Defiziten bei den Schülerinnen und Schülern (SuS) aufgedeckt wurden, ist dieser in den Fokus der Bildungsstandards gesetzt worden. Einige Ähnlichkeiten der Prozesse (wissenschaftliche Erkenntnisgewinnung und Physical Computing) und benötigter Fähigkeiten von Schülerinnen und Schülern wurden bereits von Sullivan (2008) beschrieben. Bisher fokussierte sich die Untersuchung des Physical Computing auf die verwendeten Geräte. Aufgrund der möglichen Überschneidungen des Physical-Computing-Prozesses mit einem gut erforschten Bereich in den Naturwissenschaften, sollen die Gemeinsamkeiten und Unterschiede beider Prozesse in dieser Arbeit herausgestellt werden. Insbesondere für die Informatik hat dieses Ziel einen Mehrwert, da Erkenntnisse aus der naturwissenschaftlichen Forschung möglicherweise auf die Informatik übertragbar werden und zur Grundlagenforschung in der Informatikdidaktik beitra-

gen können. Neben dem Gewinn für die Informatik sollen die Ergebnisse den Weg für einen interdisziplinären MINT-Unterricht (Mathematik, Informatik, Naturwissenschaften und Technik) unterstützen. Falls es möglich ist, die Ähnlichkeiten im Prozessbereich herauszuarbeiten, können auch bereits beschriebene inhaltliche Überschneidungen (Schulz und Pinkwart, 2015; Lindner et al., 2017) in einem interdisziplinären Unterricht mittels Physical-Computing-Geräten aufgegriffen werden. Die Idee des fächerübergreifenden und fächerverbindenden Unterrichts zieht sich als Motivation durch die vorliegende Arbeit. Das wird beispielsweise durch genutzte Modelle der Naturwissenschaften deutlich. Darüber hinaus sollen gewonnene Erkenntnisse aus dieser Arbeit interdisziplinäre Unterrichtsansätze unterstützen.
Die vorliegende Dissertation untersucht den Physical-Computing-Prozess, um auf theoretischer und empirischer Ebene die Überschneidung des Physical-Computing-Prozesses mit dem Prozess der wissenschaftlichen Erkenntnisgewinnung beurteilen zu können. Forschungsergebnisse zeigen jedoch auch, dass der wissenschaftliche Problemlöseprozess als *Forschendes Lernen* Schülerinnen und Schüler überfordern kann (Schmidt-Weigand et al., 2008). Das kann durch die *Cognitive-Load-Theory* erklärt werden, die beschreibt, dass die Kapazitäten des Arbeitsgedächtnisses begrenzt ist und die Komplexität von neuen Aufgaben deshalb reduziert werden sollte (Sweller et al., 1998). Das kann z. B. durch Materialien und Hilfestellungen realisiert werden. In der naturwissenschafts-didaktischen Forschung wurden bereits solche Hilfestellungen in verschiedenen Ausprägungen (Kirschner et al., 2006) untersucht, um Rückschlüsse auf den positiven Einfluss der Hilfestellungen auf den Erkenntnisgewinnungsprozess der SuS ziehen zu können. Um in folgenden Schritten Hilfestellungen für den Physical-Computing-Prozess entwickeln zu können, wird in dieser Arbeit eine Analyse von Problemen vorgenommen, die SuS während des Prozesses hatten.

Aufgrund des interdisziplinären Charakters dieser Arbeit, werden Begriffe verwendet, die eine unterschiedliche Bedeutung in den einzelnen Disziplinen haben können. Diese Begriffe werden im Folgenden geklärt.
Klieme (2004, S. 13) beschreibt den *Kompetenzbegriff* wie folgt: „Kompetenz stellt die Verbindung zwischen Wissen und Können her und ist als Befähigung zur Bewältigung unterschiedlicher Situationen zu sehen.“ Dabei existieren bereichsspezifische und bereichsübergreifende Kompetenzen, wobei Problemlösen in die bereichsübergreifenden Kompetenzen einzuordnen ist. Davon ist der Kompetenzbegriff von Lerninhalten abzugrenzen.
Newell und Simon (1976, S. 121) erklären den Begriff *Problem* wie folgt: „We have a problem if we know what we want to do (the test), and if we don't know immediately how to do it [...].“ Es wird eine Situation beschrieben, in der ein Zielzustand bekannt ist. Der Weg, diesen Zielzustand zu erreichen, wird als Problemlöseprozess bezeichnet.
Insbesondere im schulischen und beruflichen Kontext werden eine Vielzahl von Problemen

kollaborativ gelöst, was in der Forschung als *Collaborative-Problem-Solving* bezeichnet wird. Die Kompetenzen zum kollaborativen Problemlösen werden definiert als: „the capacity of an individual to effectively engage in a process whereby two or more agents attempt to solve a problem by sharing the understanding and effort required to come to a solution and pooling their knowledge, skills and efforts to reach that solution“ (OECD, 2017, S. 47). Durch die Zusammenarbeit von verschiedenen Agenten werden Kompetenzen benötigt, die über das individuelle Lösen von Problemen hinausgehen. Beim kollaborativen Problemlösen findet aus Sicht der Gruppenmitglieder meist die Bearbeitung eines ungenau bestimmten Problems statt, da das Verhalten der einzelnen Mitglieder nicht vorhergesagt oder kontrolliert werden kann. Diese Probleme werden als *ill-defined-Problems* bezeichnet und wie folgt definiert: „ill-defined problems may have multiple goals and underspecified problem states and actions“ (OECD, 2017, S. 51). Durch die Vielzahl von Problemzuständen und möglichen Aktionen haben diese Probleme eine besonders hohe Schwierigkeit.
Die Verwendung des Begriffs *Prozess* wird in dieser Arbeit mit dem *Problemlöseprozess* gleichgesetzt. Mayer (2007, S. 177) fasst die wissenschaftliche Erkenntnisgewinnung als einen Problemlöseprozess zusammen und erklärt ihn als *den Prozess des wissenschaftlichen Vorgehens*. Es handelt sich dabei um einen kurzfristigen Prozess und ist von projektartigen Entwicklungsprozessen abzugrenzen, wie sie von Przybylla und Romeike (2017) beschrieben werden.
Auch *Modelle* sind ein wichtiger Bestandteil der Informatik und werden in ihrer Bedeutung sehr unterschiedlich verwendet (vgl. Thomas, 2002). Mahr (2009) beschreibt die Funktionen von Modellen einerseits als „Modell für etwas“ und andererseits als „Modell von etwas“. Im Rahmen dieser Arbeit werden Modelle auch verwendet, um einen beobachteten Problemlöseprozess zu beschreiben (Modell von etwas). Sie dienen jedoch ebenfalls zur Beschreibung der Bedingungen, die der Problemlöseprozess erfüllen sollte (bspw. konkrete Phasen als Modell für etwas).

1.1 Theoretische und methodische Einordnung

Auf der lerntheoretischen Ebene ist das Forschungsgebiet Physical Computing in den Konstruktionismus einzuordnen. Diese Zuordnung wurde bereits von verschiedenen WissenschaftlerInnen vorgenommen und wird im Folgenden vorgestellt (Papert, 1980; Kafai und Resnick, 1996; Przybylla und Romeike, 2014b; Katterfeldt et al., 2015). Zunächst wird eine allgemeine Einführung in die Theorie vorgenommen. Der Ursprung des Konstruktionismus liegt in dem von Jean Piaget beschriebenen Konstruktivismus. Die etymologische Ähnlichkeit ist auf die inhaltlichen Überschneidungen zurückzuführen und ebenso auf die Tatsache, dass der Begründer des Konstruktionismus, Seymour Papert, Piagets Schüler war.

Papert charakterisierte den Konstruktionsmus wie folgt: „The word constructionism is a mnemonic for two aspects of the theory ... [of] science education [...]. From constructivist theories of psychology we take a view of learning as a reconstruction rather than as a transmission of knowledge. Then we extend the idea of manipulative materials to the idea that learning is most effective when part of an activity the learner experiences as constructing a meaningful product.“ (Papert, 1989, S. 1)
In diesem Zitat beschreibt Papert den Konstruktionismus als eine Art des Lernens durch die Umstrukturierung von Wissen sowie das haptische Konstruieren, bei dem für die Lernenden bedeutsame Artefakte geschaffen werden. Es ist z. B. für SuS anschaulicher mittels einer Schildkröte (Roboter) physikalische Gesetze zu lernen als durch einen bloßen Punkt auf einem Bildschirm. Beschrieben wird dieser Ansatz des Lernens mittels der Schildkröte wie folgt: „In LOGO environments many children have been started on the road to Turtle geometry by introducing them to a mechanical turtle, a cybernetic robot, that will carry out these commands when they are typed on a typewriter keyboard. This ‘floor Turtle’ has wheels, a dome shape, and a pen so that it can draw a line as it moves.“ (Papert, 1980, S. 56)
Entscheidend im Konstruktionismus ist jedoch, eine andere Art des Lernens zu schaffen, ohne einen Fokus auf die spezifischen Geräte zu legen. In diesem Zusammenhang wird von sogenannten Mikrowelten gesprochen, die sich die Lernenden erschaffen und erschließen. Circa zehn Jahre später wurde die Zusammenwirkung von Software und Hardware (LOGO/LEGO) von Papert und Harel (1991, S. 6) beschrieben, um den Begriff des Konstruktionismus zu klären. Dabei wird auf bestehende Projekte zurückgegriffen und implizit die Einordnung des Physical Computing in den Konstruktionismus vorgenommen: „A conceptually simple case is the addition of new elements to LEGO construction kits and to the Logo microworlds, so that children can build more ‘active’ models. For example, sensors, miniaturized computers that can run Logo programs, and motor controllers allow a child (in principle) to build a LEGO house with a programmable temperature control system; or to construct forms of artificial life and mobile models capable of seeking environmental conditions such as light or heat or of following or avoiding one another.“
Geleitet von dieser Idee entwickelten Blikstein und Wilensky das *Bifocal Modelling* und verknüpften die Computersimulation mit der Aufnahme von Sensordaten aus der realen Umgebung (Blikstein und Wilensky, 2007). Es wird das Ziel verfolgt, Modelle mit realen Daten zu testen und zu validieren. Dadurch soll ein tieferes Verständnis für wissenschaftliche Phänomene erwirkt und die wissenschaftliche Erkenntnisgewinnung unterstützt werden. Die Autoren argumentieren, dass eine Verbindung von Robotik/Sensorik mit der Computersimulation bislang fehlt, und streben diese beim *Bifocal Modelling* an. Als Lerninhalte wählten sie Wärmetransfer, Gasgesetze, chemische Reaktionen und Werkstoffwissenschaften. Sie führten mit 14 Studierenden eine Studie durch, die in zwei Testgruppen geteilt wurden. Die Studienbedingungen der Gruppen unterschieden sich darin, dass eine Gruppe nur die Computersimulation nutzte und die zweite Gruppe die Simulation mit

sensorbasierten Daten testete. Die Studierenden, die mit Sensoren arbeiteten, fühlten sich angesprochener von den Konzepten Reibung/Energieverlust, Genauigkeit, Skalen, Zeit, Koeffizienten, Skalenkonvertierung und Synchronizität im Gegensatz zu der Gruppe, die mit Computersimulationen arbeitete. Das *Bifocal Modelling* ermöglichte es Studierenden ihre Hypothesen zügig zu untersuchen sowie alternative Resultate zu beobachten. Das Debuggen von Modellen und Algorithmen ist ebenso ein wichtiges Ergebnis, das durch das *Bifocal Modelling* ermöglicht wird.

Methodisch wird in der vorliegenden Arbeit der Ansatz des *Design-Based-Research (DBR)* verfolgt, der auch als *Educational-Design-Research* bezeichnet wird. Es handelt sich dabei um eine Forschungsmethode, die insbesondere seit den 90er-Jahren an Bedeutung gewinnt. Sie untersucht Lernen an den Orten, wo es in seiner natürlichen Form auftritt, weshalb Klassenraumstudien besonders geeignet sind (vgl. Brown, 1992).
Design-Based-Research ist eine Form der empirischen Forschung in realen Lernszenarien, wobei trotzdem theoretische Erkenntnisse gewonnen werden. Somit verbindet DBR Grundlagenforschung mit angewandten Forschungsmethodiken und hat daher den Anspruch, robust sowie relevant zugleich zu sein. McKenney und Reeves (2012) beschreiben: „educational design research is a genre of inquiry – not a fixed method“ (S. 3). Sie verfolgen einen empirischen Ansatz, der iterativ und regulativ verläuft und die folgenden fünf Charakteristika enthält:

1. Theorieorientiert: Wissenschaftliche Ergebnisse werden nicht nur für die Weiterentwicklung von Theorie genutzt, sondern auch, um einen Lösungsweg für ein reales Problem zu entwickeln.

2. Interventionen befürwortend: Interventionen können verschiedene Facetten haben, wie konkrete Produkte (z. B. Lernmaterialien), Prozesse (z. B. Lehrrepertoire), Programme (z. B. Weiterbildungsangebote) oder Richtlinien (z. B. Protokolle für die Schule) (vgl. McKenney und Reeves, 2012, S. 14). Dabei besteht immer die Intention Veränderungen an der Basis zu schaffen.

3. Kollaborativ: Theorie und Praxis sollen zusammenarbeiten, so dass reale Probleme aus der Schule behandelt und konkrete Lösungen geboten werden können.

4. Reaktionsfähig: In Untersuchungen von verschiedenen realen Lehr- und Lernkontexten muss es möglich sein direkt zu reagieren, z. B. durch Veränderungen in einem Iterationszyklus.

5. Iterativ: Mehrere Zyklen von *Untersuchung*, *Entwicklung*, *Testen* und *Verändern* sind Bestandteil des DBR-Prozesses. Dabei kann eine große Studie mehrere Teilstudien enthalten, wobei jede ihren eigenen Zyklus durchläuft. Das Clustern von mehreren Studien ist daher gängig.

Die Phasen des DBR-Prozesses beschreiben McKenney und Reeves wie folgt:

1. Analysis and Exploration: In dieser Phase wird das vorliegende Problem untersucht sowie das Verständnis über das Problem entwickelt. Dabei werden ähnliche Probleme sowie deren Lösungen in Literaturreviews betrachtet. In dieser Phase werden ebenfalls ExpertInnen einbezogen, die die Praxisrelevanz in diesem Prozess sicherstellen. Dadurch kann das Problem an seinem Ursprung untersucht werden.

2. Design and Construction: umfasst die Entwicklung eines konzeptionellen Modells, um das beschriebene Problem zu lösen. Während des *Design*-Prozesses werden mögliche Problemlösungen untersucht und abgewogen, welche auf theoretischen und praktischen Grundlagen beruhen. Dabei werden ebenfalls methodische Entscheidungen getroffen. Bei der *Konstruktion* werden die Design-Ideen prototypisch umgesetzt und an den Zielzustand angepasst.

3. Evaluation and Reflection: beinhaltet das empirische Testen des entwickelten Designs und der konstruierten Interventionen. In der *Reflection* werden Abwägungen zu theoretischen Erkenntnissen und empirischen Untersuchungen getätigt. Die gewonnenen Erkenntnisse werden zur Bestätigung, Anpassung oder Widerlegung von Design-Entscheidungen genutzt.

Diese Phasen werden iterativ als sogenannte *Meso-Zyklen* durchlaufen, um ein Phänomen (*Makro-Zyklus*) zu untersuchen. Jede einzelne Phase wird als ein *Mikro-Zyklus* bezeichnet. Nach dem Durchlauf des Makro-Zyklus stehen zwei Ergebnisse im Vordergrund: konkretes Material für Anwendungen in der Praxis sowie theoretische Erkenntnisse für die Weiterentwicklung des Forschungsfelds. Insbesondere die praxisrelevanten Ergebnisse müssen im Anschluss implementiert und in einem großen Personenkreis bekannt gemacht werden (z. B. bei Lehrkräften, Dozierenden, Bildungsträgern und Herausgebern von Büchern).

DBR eignet sich für die Untersuchung von praxisnahen Forschungsgegenständen, deren theoretische Basis bislang unzureichend für praktische Anwendungen ist. Aufgrund der Flexibilität des DBR sind die Anwendungsbereiche mannigfaltig. Beispielsweise wurde im Bereich von Lehrkräftefortbildungen untersucht, welchen Einfluss diese Fortbildungen auf die Aktivierung von leistungsschwachen SuS haben (vgl. Swan, 2007). Die flächendeckende Einführung von Curricula in Schulen sowie deren Adaption für individuelle Anforderungen (vgl. Clarke und Dede, 2009) ist ein weiteres Beispiel für die Anwendung dieser Forschungsmethode.
Im Bereich der Informatikdidaktik ist DBR bislang noch rar vertreten, weshalb im Folgenden ein interdisziplinäres Beispiel im Kontext der Informatik vorgestellt wird. In diesem Projekt werden Geografiestudierende durch die Vermittlung von informatischem Wissen qualifiziert, um anschließend Tools der Geographie nutzen zu können. Die Verwendung von *Geographic-Information-Systems* (GIS) erfordert beispielsweise Programmierfähigkeiten

sowie Wissen über Datenstrukturen und Modellierung. Knobelsdorf et al. (2017) folgen dem DBR-Ansatz, indem sie basierend auf Literaturrecherchen Lernszenarios (für Studierende und Lehrende) erschaffen, implementieren und evaluieren. Dadurch können theoretische Erkenntnisse gewonnen und konkretes Lehr- und Lernmaterial entwickelt werden. In diesem Beispiel haben die AutorInnen zunächst die Fragestellung untersucht, welche wichtigen informatischen Kompetenzen Geografiestudierende benötigen, um Werkzeuge wie GIS nutzen zu können. Für die Bearbeitung dieses Problemfelds wurde ein Kurs explizit für Geografiestudierende entwickelt, der aus Seminaren, Vorlesungen, praktischen Aufgaben und einem Abschlussprojekt bestand. Der Zeitraum des Kurses erstreckte sich in der ersten Durchführung über ein Semester, wobei 18 Lehramtsstudierende an dieser fakultativen Veranstaltung teilnahmen. Abgeleitet von deutschen Informatikbildungsstandards spielten u. a. die folgenden Inhalte eine Rolle: Verständnis für Informationsprozesse in GIS; Verwendung von Datentypen für räumliche Daten; Organisation und Struktur von GIS; Präsentation und Interpretation von Diagrammen, Graphen und Resultaten, die durch GIS erzeugt wurden. Neben Themen mit einem Fokus auf informatische Kompetenzen, wurden Klima und Klimawandel in dem Kurs gelehrt. Mit diesen Inhalten wurde ein Kurs entwickelt, der inhaltlich auf die KursteilnehmerInnen zugeschnitten ist und dem pädagogischen Ansatz der Kontextualisierung folgt.
Nach dem ersten Durchlauf des Kurses befinden sich die AutorInnen zum Erscheinungszeitpunkt des Artikels im Evaluationsprozess. Erste Prognosen können sie jedoch durch Eindrücke der Kursleitung geben. Für die Thematik relevante Rückfragen der Studierenden deuten darauf hin, dass sie sich intensiv mit den Inhalten auseinandersetzten. Gemäß des DBR trägt das Projekt dazu bei, konkrete und theoretisch fundierte Lernmaterialien bzw. ein Lehrkonzept zu entwickeln. Darüber hinaus deuten die ersten Ergebnisse darauf hin, dass sich der pädagogische Ansatz der Kontextualisierung auf dieses interdisziplinäre Projekt gewinnbringend auswirkt. Somit ist der Rückschluss auf theoretische Erkenntnisse ebenfalls gegeben. Die Fertigstellung der Evaluation sowie die Überarbeitung des Kursdesigns sollen folgen, um weitere Durchläufe des Kurses vorzunehmen. In diesem Fall wird der DBR-Prozess mehrfach durchlaufen und der praktische sowie theoretische Output schrittweise adaptiert und evaluiert.
Der DBR-Ansatz scheint gewinnbringend für interdisziplinäre Forschung zu sein, in der bislang wenige theoretische Erkenntnisse vorliegen, auf denen aufgebaut werden kann. Die vorliegende Arbeit folgt in ihrer Struktur dem DBR-Prozess, was in Abschnitt 1.2 näher erläutert wird.

1.2 Struktur dieser Arbeit

Aufgrund von genannten methodischen Entscheidungen ist diese Arbeit gemäß des Design-Based-Research in mehrere Phasen gegliedert, die in einzelnen (Teil-)Kapiteln beschrieben werden. Eine Übersicht darüber wird in Abbildung 1.3 gegeben.

In Kapitel 1 wurde bereits die grundlegende Motivation dieser Arbeit dargelegt und in die Thematik eingeleitet. Ein Überblick über durchgeführte Studien, ProbandInnen und erhobene Daten wird vorab in den Untersuchungsdesigns gegeben (Kapitel 1.3).
In den folgenden Kapiteln werden im Wesentlichen drei Forschungsfragen bearbeitet. Den Ausgangspunkt bildet die Fragestellung, ob der Physical-Computing-Prozess dem Prozess der wissenschaftlichen Erkenntnisgewinnung ähnlich ist. Zur Bearbeitung dieses Forschungsfelds wird im 2. Kapitel ein Literaturreview vorgenommen, das die Phase *Analysis and Exploration I* im DBR-Prozess bildet. In dieser Phase wird das Fundament für die Untersuchung des Forschungsfelds *Physical Computing* gelegt und es werden Forschungslücken aufgezeigt. Der Literaturüberblick ermöglicht es die 1. Forschungsfrage der Dissertation herzuleiten, die wie folgt formuliert ist:

1. *Welche Gemeinsamkeiten und Unterschiede weisen die Prozesse Physical Computing und die wissenschaftliche Erkenntnisgewinnung auf? Kann Physical Computing als Arbeitsweise der Erkenntnisgewinnung in der Informatik genutzt werden?*

In Kapitel 3.2 werden das Untersuchungsdesign und damit verbundene Testinstrumente und Methoden vorgestellt, die dazu dienen, die 1. Forschungsfrage zu beantworten. Das entspricht dem Schritt *Design and Construction I* des DBR-Prozesses. Die Darstellung der Ergebnisse und die Evaluation erfolgen in Kapitel 3.3 (*Evaluation and Reflection I*).
Daraus ergibt sich die Notwendigkeit eines weiteres Literaturreviews (*Analysis and Exploration II*), das sich mit der Frage nach spezifischen Problemen (und den zugehörigen Problemursachen) von Schülerinnen und Schülern während der Arbeit mit Physical-Computing-Geräten beschäftigt. Dieses Review wird in Kapitel 4.1 vorgenommen, um die 2. Forschungsfrage zu beantworten:

2. *Welche Probleme treten bei der Interaktion mit Physical-Computing-Geräten auf und wie können diese kategorisiert werden?*

Im selben Kapitel wird das Vorgehen zur Untersuchung der 2. Forschungsfrage geplant (*Design and Construction II*) sowie die Evaluation der Daten vorgenommen (*Evaluation and Reflection II*). Da auf Daten und Methoden der 1. Forschungsfrage zurückgegriffen wird, kann die Phase *Design and Construction II* ohne umfangreiche, methodische Ergänzungen durchlaufen werden.
Basierend auf der 2. Forschungsfrage wird untersucht, wie gewonnene Erkenntnisse praktisch umzusetzen sind. Theoretische Grundlagen für die Entwicklung von analogen und digitalen Hilfestellungen zum Problemlösen werden in weiteren Literaturreviews thematisch sortiert dargelegt (Kapitel 5, 5.3, 5.1, *Analysis and Exploration III*), um die folgende Forschungsfrage zu beantworten:

3. *Welche Arten von Hilfestellungen sind effektiv für Schülerinnen und Schüler, um den Physical-Computing-Prozess zu unterstützen?*

In diesem Rahmen werden weitere Testinstrumente und Methoden verwendet, die in Kapitel 5.2 vorgestellt werden (*Design and Construction III*). Die herausgearbeiteten Ansätze für Hilfestellungen werden in Teilstudien gegliedert in den Kapiteln 5.3 und 5.4 evaluiert (*Evaluation and Reflection III*). Aufgrund der enthaltenen Teilstudien laufen innerhalb des DBR-Prozesses für die Beantwortung der 3. Forschungsfrage drei DBR-Zyklen ab, die der Entwicklung und Evaluation von Hilfestellungen dienen. Diese werden vereinfacht in Abbildung 1.3 durch den 3. Meso-Zyklus dargestellt.
Das 6. Kapitel fasst diese Dissertation zusammen und gibt einen Ausblick für weitere Forschungsansätze innerhalb der untersuchten Forschungsfelder.

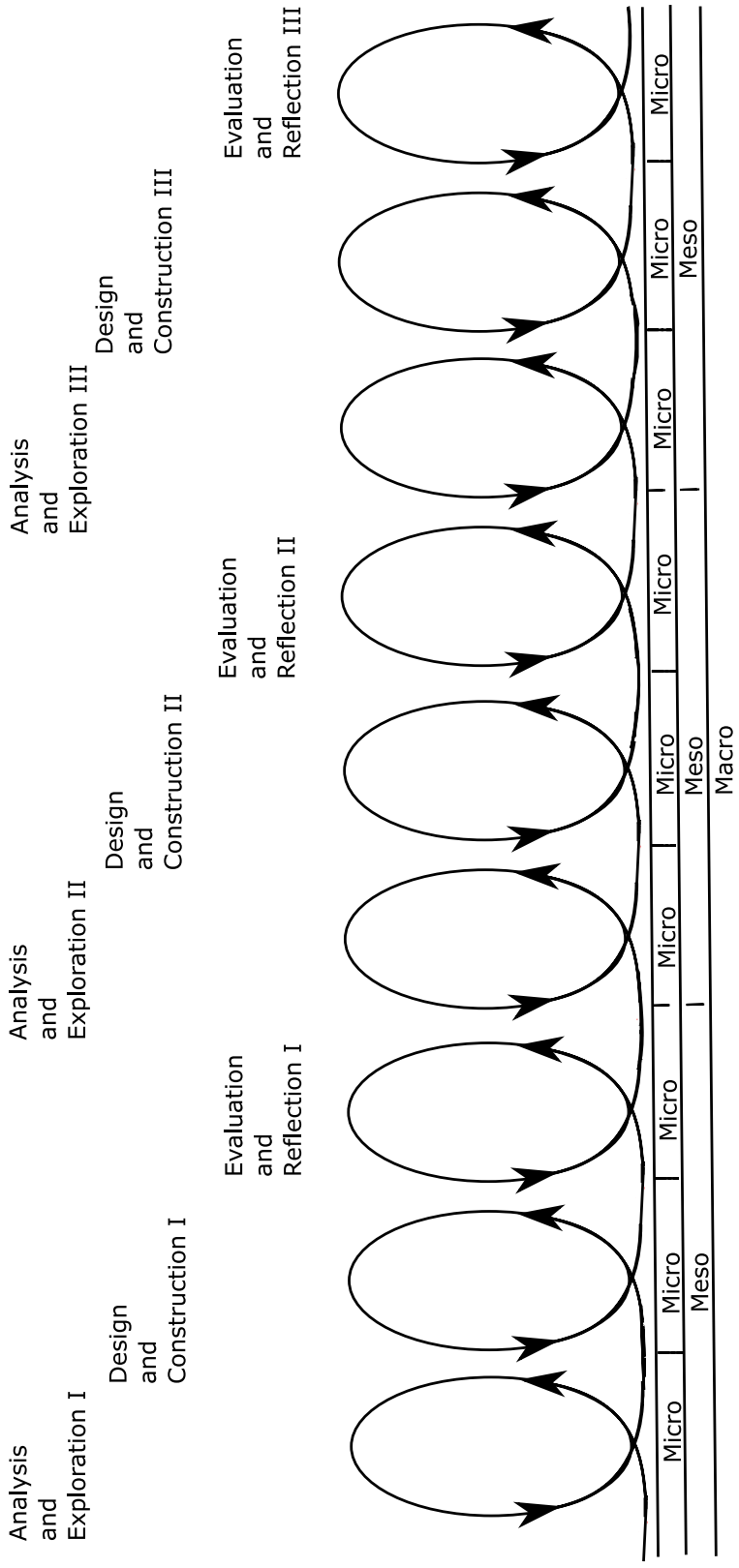


Abbildung 1.3: Struktur der Arbeit als Design-Based-Research-Prozess

1.3 Untersuchungsdesigns

In diesem Abschnitt werden zunächst die Stichproben beschrieben, aus denen sich die Studien zusammensetzen. Es wird dargelegt, welche Studiendaten zur Beantwortung der Forschungsfragen genutzt wurden und welche Physical-Computing-Geräte und Software dafür zur Verfügung standen. Im Anschluss wird ein Überblick über die Datenerhebung und die generelle Datenaufbereitung gegeben.

1.3.1 Beschreibung der Stichproben

Neben den 67 ProbandInnen für die Pilotierung des Physical-Computing-Tests nahmen weitere 45 ProbandInnen an den Studien dieser Dissertation teil. Zu diesem Zeitpunkt besuchten sie die 7. − 12. Klasse. Die SuS wurden in den folgenden Bereichen akquiriert:

- in Informatik-Ferienkursen (4 SuS, Studie 1),
- in der Schülergesellschaft Informatik der HU Berlin (14 SuS, Studie 5),
- in der Physikalischen Schülergesellschaft der HU Berlin (10 SuS, Studie 4),
- im Rahmen ihres Betriebs- oder Forschungspraktikums an der HU Berlin (17 SuS, Studien 2, 3 und 6).

Somit setzten sich die Gruppen aus SuS von verschiedenen Schulen Berlins und Brandenburgs zusammen, die Interesse an der Informatik zeigten. Die meisten SuS haben bereits durch die Schulen kurze Einblicke in die Roboterprogrammierung gewonnen, wobei sie ihre Vorerfahrungen zumeist als gering beschrieben haben. Bis auf wenige Ausnahmen hatten alle SuS bereits Programmiererfahrungen. Lediglich zwei Schüler hatten einen eigenen LEGO Mindstorms-Roboter zu Hause und viel Erfahrung damit erlangt, wobei die Interaktion mit dem Gerät mehrere Jahre zurück lag. Aufgrund des zeitlichen Abstandes sind die SuS mit einem eigenen LEGO-Roboter nicht als Experten zu bewerten. Bestünde die Gruppe von ProbandInnen aus Novizen, ist davon auszugehen, dass sie insbesondere Probleme während der Programmierphasen haben. Daher würde vorwiegend ihr Problemlösen beim Programmierenlernen erhoben werden, jedoch nicht das Problemlösen in Bezug auf das gesamte Physical-Computing-System. Bestünde die Gruppe aus ExpertInnen, wäre es schwierig Probleme und geeignete Hilfestellungen zu untersuchen, da diese SuS vermutlich gute Problemlöser wären. Es ist davon auszugehen, dass der zu untersuchende Problemlöseprozess bei ExpertInnen anders abläuft und die Probleme und Hilfestellungen für Schulgruppen nicht repräsentativ sind. Die ProbandInnen aus der Physikalischen Schülergesellschaft wurden gewählt, da bei der Arbeit mit Arduino-Mikrocontrollern physikalisches Vorwissen benötigt wird. Bei der Physikalischen Schülergesellschaft handelt es sich um interessierte und begabte SuS, die sich an der Humboldt-Universität zu Berlin regelmäßig mit Themen der Physik auseinandersetzen. Dafür besuchen sie verschiedene Forschungs- und Bildungseinrichtungen und behandeln Themen über das Schulniveau

hinaus. Alle SuS der Physikalischen Schülergesellschaft, die sich für die Teilnahme an der Studie bereiterklärten, waren an der Informatik interessiert.
In Tabelle 1.1 ist die Verteilung der Klassenstufen auf die einzelnen Studien dargestellt. Über alle Studien hinweg konzentriert sich der Altersdurchschnitt auf die Klassenstufen $9 - 10$. Die Testergebnisse von angewandten Testinstrumenten in diversen Gruppen sind in Anhang D dargestellt. Diese Zusammensetzung der ProbandInnen zeigt, dass die Studienergebnisse insbesondere für an der Informatik interessierte SuS repräsentativ sind. Es handelte sich um ProbandInnen, die an freiwilligen Kursen und Praktika an der Humboldt-Universität zu Berlin teilnahmen.

Alle Studien wurden außerhalb der Schule, entweder am Nachmittag oder in den Ferien, an der Humboldt-Universität zu Berlin durchgeführt.

1.3.2 Überblick über den Zusammenhang der Teilstudien

Im Rahmen der hier vorliegenden Arbeit wurden sechs Studien durchgeführt, die verschiedenen Forschungsfragen zuzuordnen sind. Zur Beantwortung der *1. Forschungsfrage* (Kapitel 3) wurden die ersten vier Studien durchgeführt, die ein ähnliches Setting besitzen. Studie 1 und 2 sind dabei inhaltlich identisch und werden dazu genutzt, den Prozess des Physical Computing zu analysieren (unter der Nutzung von Hilfestellung *4-Phasen analog*, Abschnitt 3.2.2). An der 1. Studie nahmen vier ProbandInnen teil und arbeiteten über zwei Tage mit LEGO Mindstorms-Robotern. Sie nutzten die zugehörige Programmierumgebung von LEGO Mindstorms EV3 und wurden am 2. Tag für mehrere Stunden videografiert. Die 2. Studie ist in der Durchführung zur 1. Studie identisch, jedoch nahmen daran fünf ProbandInnen teil. Deshalb arbeitete ein Proband ohne GruppenpartnerIn. In beiden Studien wurde der Problemlöseprozess der SuS bei der Bearbeitung von Physical-Computing-Aufgaben beobachtet.
Die 3. Studie unterscheidet sich darin, dass die SuS die Hilfestellung *Evaluationsphase digital* (vgl. Abschnitt 5.2.2) bekamen sowie den Arbeitsauftrag, ihre eigenen Beobachtungen beim Lösen der Aufgaben zu evaluieren. An dieser Studie nahmen sechs ProbandInnen teil, die zusätzlich eine Instruktion bekamen, um besser selbstständig den Problemlöseprozess zu bewältigen.
Bei der 4. Studie handelt es sich um einen ähnlichen Studienaufbau, jedoch wurden andere Physical-Computing-Geräte genutzt, um Aussagen über mehrere Geräte treffen zu können und diese zu vergleichen. Durch die Variation an Physical-Computing-Geräten soll es möglich sein, über den Bereich Physical Computing allgemeinere Schlüsse ziehen zu können. Die SuS arbeiteten mit Arduino-Mikrocontrollern und programmierten sie mit *Scratch for Arduino*. An der Studie nahmen zehn ProbandInnen aus der Physikalischen Schülergesellschaft der Humboldt-Universität zu Berlin teil. Sie erstreckte sich über zwei Tage, wobei die Intervention am 2. Tag videografiert wurde.
Die Aufteilung der 1. und 2. Studie in Teilstudien war vor allem organisatorischer Natur,

da der Zeitraum zur Studiendurchführung eingeschränkt und dadurch die Teilnehmerzahl limitiert war (für die Interventionsphasen, die paarweise nacheinander durchgeführt wurden). Bei der Durchführung der weiteren Teilstudien wurden die Altersklasse, verwendete Geräte und Hilfestellungen variiert. Das ist teilweise durch unterschiedliche inhaltliche Fragestellungen zu begründen sowie durch das Ziel, dass die gewonnen Ergebnisse einen großen Bereich des Physical Computing beschreiben sollen. Die Anzahl von interessierten SuS, die zur Teilnahme an einer Studie bereit waren bzw. sich für die Informatik interessieren, spielte dabei ebenfalls eine Rolle.

Die *2. Forschungsfrage* beschäftigt sich mit Problemen der SuS, die im Problemlöseprozess des Physical Computing auftreten. Zur Beantwortung dieser Frage wurden keine gesonderten Studien durchgeführt, sondern das Videomaterial der Forschungsfrage 1 genutzt und erneut analysiert.

In der *3. Forschungsfrage* werden Hilfestellungen beim Physical-Computing-Prozess untersucht. Eine erste Hilfestellung (*Evaluationsphase digital*) wurde bereits in der 3. Studie durch eine Unterstützung des Physical-Computing-Prozesses vorgenommen, indem die Phase der Evaluation forciert wurde.
Abgeleitet von Resultaten der 2. Forschungsfrage wurde ein Modell entwickelt, welches in der 5. und 6. Studie als Hilfestellung verwendet und untersucht wurde. Dazu wurde zunächst eine Studie als Klassenraumsituation (Studie 5) mit 14 SuS durchgeführt, die auf Grundlage des Modells Hilfestellungen bekommen und zu dem Modell befragt wurden. Diese Studie wurde über zehn Wochen (jeweils 2 Stunden an einem Nachmittag je Woche) durchgeführt, wobei in insgesamt drei Wochen Interventionen durchgeführt wurden. In den Interventionsstunden wurden Beobachtungsprotokolle angefertigt und Interviews geführt. Die SuS arbeiteten zunächst mit LEGO-Robotern und der Programmierumgebung *Open Roberta* (in den ersten drei Wochen) und danach mit Nao-Robotern, welche sie mit *Choregraphe* programmierten. Von den drei Interventionen wurde die erste Intervention mit LEGO-Robotern und zwei folgende mit Nao-Robotern durchgeführt. Im Anschluss der 5. Studie wurde das Modell der Hilfestellung adaptiert. In der 6. Studie wurde die Hilfestellung *Problemtaxonomie gestuft* genutzt, um als automatisierte Hilfestellung zu fungieren und diese zu evaluieren. Dafür wurden gleiche Geräte, Software und Aufgaben verwendet, doch die Durchführung der Studie wurde auf vier Tage kondensiert. Die Intervention mit verbundener Videografie erfolgte getrennt für die Bearbeitung der Aufgaben mit LEGO-Robotern und mit Nao-Robotern. Während der Intervention arbeiteten die ProbandInnen paarweise in einem Zeitfenster von jeweils 1 - 1,5 Stunden für jedes Gerät. An dieser Studie nahmen sechs ProbandInnen teil.
Mit den ProbandInnen der 5. und 6. Studie wurden Pre- und Posttests zum Problemlösen im Physical Computing und zu ihren Programmierkompetenzen erhoben. In der 4. Studie wurden diese nur als Pretest verwendet, da es sich um eine kurze Intervention handelte

und die Gruppe damit zur Pilotierung des Physical-Computing-Tests beitrug.
Eine Übersicht über die durchgeführten Studien, die ebenfalls detaillierte Informationen über die Altersverteilung der ProbandInnen enthält, ist in der Tabelle 1.1 zu finden.

Tabelle 1.1: Übersicht über die durchgeführten Studien und deren Zugehörigkeit zu den Forschungsfragen

	Studie 1	**Studie 2**	**Studie 3**	**Studie 4**	**Studie 5**	**Studie 6**
Forschungsfragen						
FF 1	x	x	x	x		
FF 2	x	x	x	x	x	x
FF 3			x		x	x
Testinstrumente						
BPK				x	x	x
PhC				x	x	x
Erhobene Daten						
Videographie	x	x	x	x		x
Interview					x	
Beobachtungs-protokoll					x	
Studienbedingungen						
Geräte	LEGO	LEGO	LEGO	Arduino	LEGO / Nao-R.	LEGO / Nao-R.
Software	LEGO	LEGO	LEGO	Scratch for Ard.	Open R./ Choreg.	Open R./ Choreg.
Hilfestellung	4-Pha.	4-Pha.	Eval-Pha.	4-Pha.	ProbTax.	ProbTax.
Dauer in Tagen / Wochen	2 T	2 T	2 T	2 T	10 W	4 T
Gruppenzusammensetzung						
Anzahl	4	5	6	10	14	6
weiblich / männlich	4/0	2/3	2/4	5/5	2/12	1/5
Klasse (Anzahl)	-	-	-	-	7 (5)	-
	-	-	-	-	8 (4)	-
	-	9 (5)	9 (4)	-	9 (4)	9 (4)
	10 (4)	-	-	10 (1)	10 (1)	10 (1)
	-	-	11 (2)	11 (3)	-	11 (1)
	-	-	-	12 (6)	-	-

1.3.3 Allgemeiner Ablauf der Datenerhebung

Bei allen Datenerhebungen haben zunächst Übungsphasen stattgefunden, die den Umgang mit den Geräten und den Programmierumgebungen zum Ziel hatten. Dadurch sollte vermieden werden, dass das zu beobachtende Verhalten und die Prozesse durch Schwierigkeiten im Umgang mit dem Material beeinflusst werden. In den videografierten Interventionsphasen wurden die ProbandInnen paarweise in einen Raum für die Durchführung der Laborstudie gebeten.
Alle erhobenen Daten sind durch Studien mit SuS der Sekundarstufe I und II erhoben worden, die freiwillig an den Studien teilnahmen. Alle Angebote wurden an der Humboldt-Universität zu Berlin außerhalb des Schulbetriebs durchgeführt. Zur Rekrutierung der SuS für Workshops und der Schülergesellschaft Informatik wurden Schulnetzwerke der Humboldt-Universität zu Berlin genutzt. Die individuelle Bewerbung von SuS war in jedem Fall möglich und wurde genutzt. Informationen zum Datenschutz sowie die Einverständniserklärungen zur Teilnahme an der Studie wurden im Vorfeld gegeben bzw. eingeholt.

1.3.4 Datenaufbereitung

Die Videos wurden im Anschluss der Studien transkribiert (dazugehörige Regeln siehe Anhang E). Das geschah zum Großteil durch die Studienleitung, teilweise mit Unterstützung durch zwei studentische Hilfskräfte. In dem letzten Fall wurden die Transkripte von der Studienleitung durch die Sichtung der Videos gegengelesen.
Die angefertigten Protokolle der 5. Studie wurden von Studierenden des Seminars zur Schülergesellschaft angefertigt. Diese waren zugleich die betreuenden Personen, die Hilfestellungen in den Interventionsstunden leisteten. Die Transkription der Audiodateien von den zugehörigen Interviews wurde von einer studentischen Hilfskraft vorgenommen.
In der 6. Studie wurden von der Hilfestellung leistenden Person (Wizard), Screenshots von den Programmen der SuS angefertigt. Des Weiteren verschickte diese Person Hinweise mit einem Messenger-Dienst, die ebenfalls von dieser Person protokolliert wurden.

Kapitel 2

Theoretische Grundlagen

In diesem Kapitel werden wesentliche Aspekte des Forschungsfelds Physical Computing vorgestellt und das Themenfeld abgegrenzt. Dafür werden zunächst verschiedene Begriffsdefinitionen betrachtet und Forschungsergebnisse präsentiert, die den State-of-the-Art des Physical Computing darlegen. Die aktuelle Rolle von interdisziplinären Projekten, die mittels Physical Computing realisiert werden, wird durch einige Beispiele veranschaulicht. Dem schließt sich eine Literaturrecherche im Bereich der wissenschaftlichen Erkenntnisgewinnung an. Die Abgrenzung verschiedener Begriffe und die Vorstellung eines Prozessmodells zum Experimentieren stehen dabei im Fokus.
Diese Literaturauswahl bildet den 1. Mikro-Zyklus (Analysis and Exploration I) im DBR-Prozess und dient der Erschließung des Forschungsfelds. Am Ende des Kapitels werden von identifizierten Forschungslücken konkrete Forschungsfragen für diese Dissertation hergeleitet.

2.1 Stand der Forschung im Physical Computing

Der Begriff des *Physical Computing* hat sich in den letzten Jahren herausgebildet und wird generell aus verschiedenen Blickwinkeln betrachtet. Es handelt sich dabei einerseits um einen Term, der offen genug sein muss, um neue Geräte einzuschließen, andererseits ist er allgemeingültig und nur schwach abgegrenzt gegenüber anderen Begriffen. Physical Computing beschreibt eine Vielzahl verschiedener Technologien, die bereits zuvor existierten und mit anderen Terminologien beschrieben wurden. So sind z. B. Roboter Objekte des Physical Computing, obwohl der eigenständige Wissenschaftszweig der Robotik existiert. O'Sullivan und Igoe grenzen den Begriff hingegen von der Robotik ab, da Roboter zumeist nur kurzzeitig mit ihrer Umgebung interagieren. Physical-Computing-Systeme interagieren üblicherweise längerfristig mit der Umgebung. Sie beschreiben Physical Computing als „[it] is about creating a conversation between the physical world and the virtual world of the computer"(O'Sullivan und Igoe, 2004, S. xix).
Przybylla und Romeike wählen einen ähnlichen Ansatz, in dem sie die Interaktion der

Geräte mit der Umgebung detaillierter beschreiben: „Interactive Objects [...] perceive their environment with sensors, which in turn deliver data to be processed by the microcontroller. According to the configuration of the systems these data are processed and passed on to the actuators“ (Przybylla und Romeike, 2014a, S. 9). Die Aufnahme von Umgebungsdaten geschieht demnach durch Sensoren (wie z. B. Berührungs-, Ultraschall- und Lichtsensoren). Aktuatoren (wie z. B. Motoren und LED-Lampen) beeinflussen die Umgebung und greifen aktiv in sie ein.
Cross et al. hingegen verwenden Robotik synonym zu Physical Computing und schließen ebenfalls Roboter neben weiteren Mikrocontrollern unter *Robotics* ein (Cross et al., 2015). In dieser Arbeit wird der Begriff des Physical Computing von Przybylla und Romeike verwendet, da dieser die Charakteristiken von Physical-Computing-Geräten angemessen beschreibt. Aufgrund der darin beschriebenen Systemkomponenten von Physical Computing, wird der Begriff Physical-Computing-System synonym zum Physical Computing verwendet. Zusätzlich wird jedoch die Gerätevielfalt von Cross et al. aufgegriffen, die die Zugehörigkeit von Robotern zum Physical Computing impliziert. Literaturbasiert kann dieses Vorgehen damit begründet werden, dass der Physical-Computing-Begriff (vgl. Przybylla und Romeike, 2014a; O'Sullivan und Igoe, 2004) ebenfalls Charakteristika von Robotern beschreibt. Somit werden Roboter in dieser Arbeit als Physical-Computing-Geräte eingeordnet. Wird im Folgenden eine Unterscheidung der Geräte vorgenommen und explizit die Verwendung von z. B. Robotern beschrieben, so wurden in diesem Kontext ausschließlich Roboter und keine weiteren Physical-Computing-Geräte verwendet. Die Unterscheidung der Geräte kann wichtige Informationen bieten, die für weitere Kategorisierungen in der Physical-Computing-Forschung notwendig sind.

Die Abgrenzung des zuvor beschriebenen *Bifocal Modelling* zum Physical Computing besteht darin, dass beim *Bifocal Modelling* zwar Daten aus der Umgebung aufgenommen und verarbeitet werden, jedoch der Eingriff in die Umgebung fehlt, um von einer Interaktion mit der Umgebung zu sprechen. Da es sich bei diesem Ansatz um Kontexte der Naturwissenschaften handelt, ist das Vorgehen vertretbar. Jedoch zeichnen sich Modelle in der Informatik dadurch aus, dass sie ein Feedback an die Umgebung zurückgeben und diese dadurch erneut beeinflussen.
Einen Abriss zur Entwicklung des Physical Computing und dazu gehörenden Geräten nimmt Blikstein vor. Er stellt heraus, dass die Geräte inzwischen erneut besonders technisch werden – wovor die NutzerInnen zuvor geschützt werden sollten. Dadurch stellt sich der Prozess als zeitintensiv heraus, der benötigt wird, um die Geräte funktionsfähig zu machen (Blikstein, 2013).

Przybylla und Romeike (2015) nahmen eine Einteilung verschiedener Geräte im Physical Computing vor. In dieser Klassifikation werden verschiedene Typen von Physical-Computing-Geräten nach ihrer Komplexität eingeteilt und eine weitere Unterteilung ihrer

Funktionalität und des Aufbaus vorgenommen (siehe Tabelle 2.1). In diesem Modell sind die LEGO Mindstorms-Roboter relativ weit oben angesiedelt, was bedeutet, dass sie in der Komplexität als vergleichsweise einfach eingestuft werden. Mikrocontroller wie Arduinos hingegen, weisen eine höhere Komplexität auf und werden tiefer in der Übersicht angeordnet. Aus der Tabelle geht ebenfalls hervor, ob die Geräte in Modulen angelegt sind oder alle Funktionalitäten bereits *on-Board* und damit fest verlötet sind. Diese Einteilung verbleibt auf einer technischen Ebene, was einerseits Bliksteins Einschätzung der zunehmenden technischen Komplexität widerspiegelt. Andererseits enthält das Modell keine Informationen über Anwendungsbereiche von Physical Computing im Informatikunterricht oder Hinweise, die auf den ablaufenden Problemlöseprozess während der Interaktion mit Physical-Computing-Geräten schließen lassen.

Tabelle 2.1: Klassifikation von Physical-Computing-Geräten nach Przybylla und Romeike (2015)

Difficulty	**Type**	**Features**		**Examples**
easy	programmable toys			Finch, Beebot, Big Track
	programmable bricks (modular)			LEGO WeDo, LEGO Mindstorms, Pico Cricket
	I/O devices	everything on board with modules		Phidges, Theremino, Senseboard Kit
		everything on board without modules	*sensing*	PicoBoard, HCIs (e.g. Makey-Makey)
			acting	-
			sensing and acting	SenseBoard
		nothing on board		Velleman Board
	micro controller boards	with modules		MyIG, Tinkerkit, Hummingbird, Gadgeteer
		without modules		Arduino Family, Wiring Board, E-Textiles
	mini computers	with modules		Arduino TRE / Intel Galileo + Arduino-compatible modules, Raspberry Pi + PiFace
complex		without modules		Raspberry Pi, Beagle Board, Intel Galileo

Die Einteilung von Physical-Computing-Geräten wurde ebenfalls auf inhaltlicher Ebene vorgenommen und beschreibt die Verwendung der Geräte. Katterfeldt et al. (2015) motivieren ihre Arbeit damit, dass inzwischen eine große Vielfalt an Geräten sowie sogenannte *Construction Kits* (Physical-Computing-Baukästen) existieren, die derzeit für ihre Anwendungsfälle nicht ausreichend strukturiert sind. Daraufhin entwickelten sie eine Taxonomie,

basierend auf existierender Theorie und Workshops mit verschiedenen Baukästen. Auf

Tabelle 2.2: Taxonomie von Kit-Eigenschaften und Anwendungsfällen nach Katterfeldt und Dittert (2017)

Merkmal er**W**ünscht o. er**F**orderlich für Anwendungsfall	Geschichtenerz.	Spieldesign	Intel. Objekte	Auton. Wesen	Erf. körperl. Akt.	naturw. Exp.
Alleinstehend	F		F	W	F	W
All-in-one		W				
Einbettung			F		F	W
Wearable	W				F	
USB-client		F				
Sensorvielfalt	W		W	W	F	F
Motorik	F			F		
Zahlreiche I/O			W		W	W
kabellose Konnektivität	W		W		W	W
Programmierbarkeit	W	W	W	W	F	F
Datenspeicherung						W

den Achsen der Tabelle 2.2 sind Anwendungsfälle und Anwendungskategorien dargestellt, die drei verschiedene Relationen zueinander haben können. Eine Anwendungskategorie kann Merkmale aufweisen, die für einen Anwendungsfall *erwünscht* (W) sind, *erforderlich* (F) sind oder *keine* Relation zueinander aufweisen. Die Anwendungsfälle sind untergliedert in Geschichtenerzählen, Spieldesign, intelligent Objekte, autonome Wesen, Erfassung körperlicher Aktivität und wissenschaftliche Experimente. Die Anwendungskategorien beschreiben diverse Merkmale, die für die Realisierung von Projekten benötigt werden. Sie sind gegliedert in: alleinstehend, all-in-one, Einbettung, Wearable, USB-Client, Sensorvielfalt, Motorik, zahlreiche I/O, kabellose Konnektivität, Programmierbarkeit und Datenspeicherung.

An dieser Stelle werden zwei Anwendungsfelder tiefer gehend betrachtet. Das Feld *autonome Wesen* umfasst Roboter und andere Fahrzeuge, die zu autonomem Verhalten fähig sind. *Naturwissenschaftliche Experimente* ist auf die Erstellung von Messgeräten durch die Aufnahme und Verarbeitung von Umgebungsdaten durch Sensoren fokussiert. Als erforderliche Merkmale werden für diese Anwendungsfelder die *Motorik* (die Ausstattung mit Motoren), die *Sensorvielfalt* (das Vorhandensein diverser Sensoren) und die *Programmierbarkeit* (Veränderbarkeit durch selbst geschriebene Programme) angegeben. Erwünschte Merkmale sind zusätzlich, dass die Geräte *alleinstehend* verwendet werden können, d. h. ohne den dauerhaften Anschluss an einen Computer. Insbesondere für naturwissenschaftliche Experimente ist es von Vorteil, wenn die Geräte in andere physische Objekte *eingebettet* werden können. Damit geht einher, dass *zahlreiche I/O*-Anschlüsse vorhanden sind, um die Sensoren mit einem Gerät zu verbinden. Eine *kabellose Konnektivität* zur Kommunikation mit anderen Geräten und die *Datenspeicherung* haben für *naturwissenschaftliche Experimente* eine besondere Bedeutung. Abgeleitet von diesen Anforderungen, kann ge-

schlussfolgert werden, dass sich Arduino Uno-Mikrocontroller für naturwissenschaftliche Experimente gut eignen. Mit dem Anschluss von Sensoren können die meisten Anforderungen erfüllt werden. Die kabellose Konnektivität, Datenspeicherung und die alleinstehende Verwendung hängen jedoch mit nötigen Zusatzmodulen (z. B. ein WLAN-Modul) und der verwendeten Programmierumgebung zusammen. Abhängig von den konkreten Experimenten und organisatorischen Rahmenbedingungen scheinen vor allem die letztgenannten Anforderungen vernachlässigbar zu sein. LEGO Mindstorms-Roboter können durch ihre Sensorvielfalt und Programmierbarkeit ebenfalls für naturwissenschaftliche Experimente genutzt werden und eignen sich hervorragend für die Aufgaben im Bereich von autonomen Wesen. Nao-Roboter bieten eine große Anzahl von Motoren, weshalb komplexe Bewegungen erzeugt werden können. Aufgrund der Kameras und Geräuschsensoren, können auch mit ihnen naturwissenschaftliche Experimente durchgeführt werden.
Diese Taxonomie eignet sich besonders gut für den Bereich Physical Computing, da sie nicht spezielle Geräte adressiert, sondern deren Spezifikationen. Für eine Übertragbarkeit in die Schulpraxis ist diese weiterhin technisch ausgerichtete Einteilung womöglich zu allgemein. Die aufgezeigten Anwendungsszenarien zeigen eine Bandbreite an Physical-Computing-Unterrichtsgegenständen auf, wobei Beschreibungen von konkreten Interaktionen der SuS mit den Geräten offen bleiben. Da diese Taxonomie auf Erfahrungen in Workshops basiert, ist schwer nachzuvollziehen, wie die Autorinnen zu den Entscheidungen gelangt sind.

In den letzten Jahren wurden verschiedene Untersuchungen bezüglich der Motivationsförderung bei SuS durch Physical-Computing-Geräte, zumeist mit LEGO-Robotern, durchgeführt. Dabei kamen diese zu unterschiedlichen Ergebnissen. Kaloti-Hallak et al. haben die intrinsische und extrinsische Motivation, die Selbstwirksamkeit und die Selbstbestimmtheit bei der Teilnahme an dem Robotik-Wettkampf *First Lego League* von 13- bis 15-Jährigen untersucht (Kaloti-Hallak et al., 2015a). Diese Studie zeigte keine signifikanten Veränderungen in den Bereichen zwischen dem Beginn und dem Ende der Intervention. Die AutorInnen interpretieren dies als positives Zeichen, da die Einstellung und Motivation von Beginn an hoch waren und es bis zum Ende blieben. Sie erläutern, dass diese Werte wahrscheinlich wegen der Begeisterung der Lernenden, neue Technologien nutzen zu können sowie durch die Wettkampfteilnahme, hoch waren. Innerhalb der Aktivitäten flachte die Begeisterung ab, da der Roboter durch diverse Einflüsse bei gleichem Input unterschiedliches Verhalten zeigen kann. Des Weiteren erzielten die SuS nicht das gewünschte Ergebnis bei einem mit dem Projekt verbundenen Wettkampf. Zum Abschluss war ein Anstieg von Einstellung und Motivation zu beobachten, wobei auffällig ist, dass die Mädchen am Ende eine positivere Einstellung und Motivation aufwiesen als zu Beginn. Eine weitere Studie zeigte, dass Lernende durch die Arbeit mit Robotern einen positiven Einfluss auf die Lernmotivation beschrieben, aber auch eine mögliche Spezialisierung in einem natur- oder ingenieurwissenschaftlichen Bereich erwägten (Verner und Ahlgren, 2004). Den positiven

Einfluss der Robotik auf die Einstellung von SekundarschülerInnen wird auch von weiteren WissenschaftlerInnen berichtet (Miller und Stein, 2000). Lauwers et al. haben ähnliche Untersuchungen auf der universitären Ebene unternommen. Sie untersuchten Kurse, die Roboter zum Lerngegenstand hatten und verglichen sie mit Modulen ohne Roboter. Sie fanden einen signifikanten Anstieg in der positiven Einstellung und Motivation von Studierenden beim Lernen mit Robotern. Allerdings konnten keine signifikanten Unterschiede in der Nachhaltigkeit des Gelernten gefunden werden (Lauwers et al., 2009). Auf der universitären Ebene untersuchte eine ähnlich gelagerte Studie zwei Gruppen, die mit und ohne LEGO-Robotern lernten. Dabei konzentrierten sich die Autoren unter anderem auf die intrinsische und extrinsische Motivation, Selbstwirksamkeit und Testangst. Eine signifikante Veränderung konnte gemessen werden. Die Gruppe, die mit den LEGO-Robotern arbeitete, zeigte einen signifikanten Rückgang der extrinsischen Motivation (McWhorter und O'Connor, 2009). Diese Entwicklung wird von den Autoren positiv bewertet, da der Rückgang der extrinsischen Motivation auf einen Anstieg der intrinsischen Motivation hindeuten kann. Ryan und Deci (2000) beschreiben „intrinsisch motiviertes Verhalten, das auf Interesse beruht und die immanenten psychologischen Bedürfnissen von Kompetenz und Autonomie erfüllt, als Grundmuster für selbstbestimmtes Verhalten. [...] Durch die Prozesse der Verinnerlichung und Integration ist es möglich, dass extrinsisch motiviertes Verhalten selbstbestimmter wird“ (S. 65, übersetzt durch Autorin). Somit wird die Schlussfolgerung von McWhorter und O'Connor (2009) unterstützt.

Ein weiterer Ansatz zur Verwendung von Physical Computing verfolgt das Ziel der Förderung von Kreativität. Sentance und Schwiderski-Grosche untersuchten dies bei 11- bis 18-jährigen SuS und konnten feststellen, dass der Umgang mit *.NET Gadgeteer* die SuS unterstützte kreativ tätig zu werden. Des Weiteren schätzten die SuS, dass dieser Unterrichtsgegenstand neue Herausforderungen im Vergleich zu weiteren Feldern des britischen Curriculums beinhaltet (Sentance und Schwiderski-Grosche, 2012). Ebenfalls zur Schaffung von kreativen Lernumgebungen sowie zur Förderung von Interesse und Motivation für die Informatik wurde ein normatives Curriculum entwickelt (Przybylla und Romeike, 2014b). Dieses hat den Anspruch, Physical Computing in verschiedene Bereiche des Informatikunterrichts zu bringen und zeigt mögliche Verknüpfungspunkte auf. Die AutorInnen befassen sich mit den Thematiken *eingebettete Systeme*, *Modellierung und Implementation* und *digitale Repräsentation von Informationen*, die sie inhaltlich detaillierter aufgliedern. Sie kommen zu dem Schluss, dass Physical Computing viele Themenbereiche der Informatik adressiert und sich eignet, um die theoretische und technische Informatik, aber auch die Einflüsse auf die Gesellschaft adäquat zu vermitteln. Der Beitrag basiert auf der Untersuchung existierender Curricula im Bereich des Physical Computing (vgl. Qiu et al., 2013; Carnegie Mellon Robotics Academy, 2014) und normativer, nationaler und internationaler Bildungsstandards der Informatik (vgl. Gesellschaft für Informatik e.V., 2008; Tucker, 2004). Es wurden Schnittstellen herausgearbeitet, bei denen sich Physical

Computing als Lerngegenstand eignet, um Aspekte der Informatik zu vermitteln. Diese wurden durch Unterrichtsbeispiele belegt. Jedoch geht aus dem von Przybylla und Romeike (2014b) entwickelten Curriculum nicht hervor, welche Bestandteile deskriptiver Natur sind, aus normativen Quellen stammen oder auf Erfahrungen basieren. Angewandte Regeln für eine Entscheidung, wann und warum sich Physical Computing als Lerngegenstand eignet, bleiben ebenso aus wie eine systematische Evaluation des aufgestellten Curriculums. Nichtsdestotrotz handelt es sich um eine umfassende Grundlage zur Veranschaulichung von Einsatzmöglichkeiten und Potentialen von Physical Computing im Informatikunterricht.
Den Lernzuwachs von Informatikkonzepten durch Robotikkurse untersuchten Kaloti-Hallak et al. im Rahmen der First Lego League bei 13- bis 15-Jährigen. Dabei evaluierten sie die informatischen Konzepte: Input/Output und Datenaustausch mit Sensoren. Analysiert durch die Bloomsche Taxonomie, konnten sie zeigen, dass fast alle SuS sinnerfülltes Lernen zeigten. Außerdem konnten einige SuS höhere Stufen in der Bloomschen Taxonomie erreichen als zuvor (Kaloti-Hallak et al., 2015b).

Inzwischen gibt es verschiedene Ansätze, wie Physical Computing in den Unterricht gebracht werden kann. Kafai et al. behandeln das Thema Physical Computing in Form von Textilien, die sie mit Mikrocontrollern, Sensoren und Aktuatoren vernähen. Sie verwenden diese, um Kernkonzepte der Informatik zu vermitteln und die existierenden Vorstellungen zur Informatik zu erweitern (Kafai et al., 2014b). Durch die Verwendung eines Arduino LilyPads ist es möglich Physical-Computing-Geräte unscheinbar in Kleidung und Accessoires einzubauen, die eine ähnliche Funktionalität besitzen wie Arduino Uno-Mikrocontroller. In der Studie analysierten Kafai et al. die Struktur und Funktionalität der konstruierten Physical-Computing-Objekte und den dazugehörigen Programmcode, der von 15 SuS im Alter von 16 - 18 Jahren in einem zehnwöchigen Kurs erarbeitet wurde. Des Weiteren wurde der gewählte Designansatz und das anschließende Debuggen der sogenannten *E-Textilien* untersucht. Frauen und ethnische Minderheiten (wie AfroamerikanerInnen, LateinamerikanerInnen und amerikanische UreinwohnerInnen) sind bereits in der schulischen Ausbildung im Fach Informatik unterrepräsentiert oder der Zugang zum Material fehlt (Wilson et al., 2010). Die Motivation des E-Textilien-Ansatzes ist, diesen Zustand zu verändern, indem ein größerer Personenkreis durch das verwendete Material begeistert und der Zugang zum Material finanziell erleichtert wird. Es wird argumentiert, dass bei dem Projekt Materialien genutzt werden, die in der Historie vor allem von Frauen verarbeitet wurden.
Kafai et al. beobachteten, dass die Wiederverwendung und Vermischung von Codeteilen sehr üblich beim Umgang mit E-Textilien ist. In der Publikation schreiben sie (Kafai et al., 2014b, S. 1:15) „debugging e-textiles is a complex process, more than debugging program code, because bugs can be caused by code, circuit design, or crafting“. Sie stellen heraus, dass viele SuS Probleme beim Debuggen hatten. Dies erklären sie mit der Komplexität des

Prozesses, da die Problemursache unklar ist, und auf den Ebenen von Schaltkreisen, der Programmierung oder der Produktherstellung/Handarbeit liegen kann. Hinzu kommt, dass die Projekte sehr unterschiedlich sind und dadurch viele individuelle Probleme entstehen. Die Vorstellungen der SuS konnten bei dem Projekt beeinflusst werden, so dass die SuS ihr Bild über die Informatik erweiterten und in ein realistischeres, für sich persönlich relevantes Bild wandelten. Eine weitere Beobachtung ist, dass die SuS den Hands-on-Charakter des Materials sehr schätzten und dass sie die einzelnen Komponenten des Systems anfassen konnten. Folgende Textpassage deutet auf eine hohe Wertschätzung der SuS hin: „Trinity liked that e-textiles took programming 'outside of [the] computer,' while Megan pointed out, 'we can touch it, we can feel it, we know what's going on'" (S. 1:17). Des Weiteren wurde das Selbstkonzept der SuS über sich als potentielle WissenschaftlerInnen erweitert und Informatik als Themenfeld weiter erschlossen. Ein großer Erfolg ist, dass Mädchen wie Jungen gleichermaßen in den verschiedenen Bereichen der Handarbeit, sowie der Schaltkreiskonstruktion und des Programmierens tätig waren. Kafai et al. gehen in dieser Veröffentlichung auch auf zwei Hauptproblemursachen der SuS ein. Das Herstellen von Schaltkreisen mit Faden und Stoffen bereitete viele Schwierigkeiten sowie die Programmierung von den physisch gelegten Stromkreisen. Die Autorinnen erläutern, dass die Verbindung von Textilverarbeitung mit elektrischen Geräten (durch Basteln mit der Elektronik und dem Programmieren) die innere Funktionalität von Schaltkreisen und Programmierung sichtbar macht. Es wird der Schluss gezogen, dass das Sichtbarmachen von Technologien unter bestimmten Umständen förderlich für das Lernen sein kann. Es werden gezielt zwei Bereiche miteinander verbunden, die zuvor eher mit Frauen (Handarbeiten) und Männern (Ingenieurswissenschaften) verknüpft waren (Kafai et al., 2014a).

In der Literatur wurden in den letzten Jahren vor allem Physical-Computing-Geräte und zu lernende Inhalte beschrieben. Bislang mangelt es z. B. an Untersuchungen und Konzepten, wie mit programmierbaren Kits tiefgehendes Lernen über digitale Technologien gefördert werden kann (Katterfeldt et al., 2015, S. 8). Basierend auf vielen Workshops leiteten Katterfeldt et al. drei Leitideen ab, die Physical-Computing-Lernumgebungen gewährleisten sollten, um ein tiefgehendes Lernen zu begünstigen. Diese werden beschrieben als:

1. *Begreifbarkeit*: „Be-greifbarkeit stands for bringing body, mind, abstract model and concrete interface together by means of tangible digital media to allow for reflection as a prerequisite for deep and sustainable learning".
   Es wird eine intensive Auseinandersetzung und Reflexion mit den Geräten beschrieben, die den Körper, Geist, abstrakte Modelle und eine konkrete Schnittstelle inkludieren. Dafür sollen Lernszenarien geschaffen werden, die ein iteratives Vorgehen unterstützen und den SuS Zeit und Raum geben ihr Vorhaben umzusetzen, ohne in einer prototypischen Entwicklungsphase zu verbleiben.

2. *Imagineering*: „Imagineering connotes the reciprocal process of referring to personal meaningful ideas, or imaginations, and implementing these using technology for digital fabrication“.
   Diese Idee adressiert die persönliche Relevanz, die das Anfertigen von Artefakten oder bereits bestehende Materialien für SuS haben kann. Als Beispiel für eine Anwendung mit einer Bedeutung für die Lebenswelt der SuS, nennen die Autorinnen die Verbesserung von Fußballschuhen durch das Einbauen von LED-Lampen.

3. *Self-efficacy*: „Self-efficacy refers to being in control of the digital medium, but also setting oneself and ones capabilities in relation to the technology and its limitations“.
   Die SuS sollen in die Rolle der Produktion von Artefakten versetzt werden und dadurch ebenfalls die Grenzen der Geräte kennenlernen. Sie müssen ihre Vorstellungen anpassen, wenn das Artefakt auf eigenen Ideen basiert.

Die aufgezeigten Prinzipien sollen Lehrende und Forschende dabei unterstützen weitere Unterrichtsszenarien im Bereich Physical Computing zu entwickeln und umzusetzen. Dabei wird das Ziel verfolgt, tiefgründigen Lernerfolg zu erreichen.
Die Kritik an der bislang gerätezentrierten Forschung innerhalb des Physical Computing wird in einer Befragung von Lehrkräften verdeutlicht. Sentance et al. begleiteten Lehrkräfte in England bei der Einführung des *BBCmicro:bit*. Die Autorinnen fassen eines der Interviews wie folgt zusammen: „One teacher complained that they had not been consulted and included in the development of the product, roll-out and resources. Giving a device to pupils is meritorious but the support and engagement of teachers is also a key factor“ (Sentance et al., 2017, S. 93). In diesem Zitat wird deutlich, dass viele Geräte für Lernkontexte bereitgestellt werden, jedoch kaum mit didaktisch fundierten Konzepten. Weitere Lehrkräfte bemängelten, dass Lehrkräfte ohne ausreichende Unterstützung unfähig seien die Probleme der SuS zu beheben und sich in einer großen Abhängigkeit zu den bereitgestellten Tutorien und Materialien befinden. Darüber hinaus benötigen sie eine Absicherung über logistische und technische Hilfestellungen, damit ein reibungsloser Ablauf der Unterrichtsstunden gewährleistet werden kann.
Es existieren vermehrt Hinweise, die auf einen positiven Einfluss von der Arbeit mit LEGO Mindstorms-Robotern auf die intrinsische Motivation von SuS hindeuten. Insbesondere der Hands-on-Charakter des Physical Computing bietet dabei neue Zugänge und wird als besonders ansprechend für Mädchen und Frauen bewertet. Dennoch handelt es sich beim Physical Computing um eine komplexe Tätigkeit, dessen Hürden beschrieben werden. Es werden konkrete Problemursachen benannt, die auf verschiedene Physical-Computing-Geräte übertragbar sind. Das hier gewählte Beispiel der E-Textilien verdeutlicht, dass die Lebenswelt von SuS adressiert wird und Physical-Computing-Geräte in vielen verschiedenen Bereichen Anwendung finden können. Jedoch darf die Physical-Computing-Forschung nicht auf der Ebene von Motivationsförderung verbleiben. Bislang wird Physical Computing genutzt, um neue Zugänge u. a. zur Informatik zu schaffen. Der Einstieg in das Phy-

sical Computing ist besonders niedrigschwellig, das Physical-Computing-System jedoch trotzdem komplex. Insbesondere für SuS sowie Lehrkräfte, die bislang keinen Kontakt zu Physical-Computing-Geräten hatten, sollten konkrete Hilfestellungen geboten bekommen, um mit auftretenden Problemen umgehen zu können. Andernfalls besteht die Gefahr, dass die Lernenden und Lehrenden durch anhaltende Frustration das Interesse am Physical Computing verlieren.

Ein weiterer Aspekt des Physical Computing ist bislang kaum erforscht: der konkrete Prozess, der bei der Arbeit mit diesen Geräten abläuft, und die Möglichkeit der Steuerung des Physical-Computing-Prozesses. Resnick und Rosenbaum beschreiben verschiedene existierende Ansätze innerhalb der Maker Szene, die unterschiedliche Einflüsse auf das Lernen haben können. Im Detail gehen sie auf *Tinkering* ein und beschreiben es als Arbeitsweise wie folgt: „tinkering as a valid and valuable style of working, characterized by a playful, exploratory, iterative style of engaging with a problem or project. When people are tinkering, they are constantly trying out ideas, making adjustments and refinements, then experimenting with new possibilities, over and over and over“ (Resnick und Rosenbaum, 2013, S. 2). Dabei beschreiben sie einen Problemlöseprozess, der iterative und experimentelle Züge besitzt, jedoch im Ziel flexibel bleibt. Tinkering beschreibt das ungeplante Vorgehen und Problemlösen bei der Interaktion mit Physical-Computing-Geräten. Bei der Prozessbeschreibung wird ein Experimentierbegriff genutzt, der sich von dem in den Naturwissenschaften unterscheidet. Im naturwissenschaftlichen Kontext stellen Gyllenpalm und Wickman heraus, dass der Term *Experiment* synonym zum *kontrollierten Experiment* verwendet wird. Sie definieren das *kontrollierte Experiment* wie folgt: „In the context of scientific inquiry, a ‘controlled experiment’ is a particular kind of research method on dependent and independent variables, and which involves the testing of hypotheses (tentative explanations) by deriving predictions that can be compared with empirical evidence“ (Gyllenpalm und Wickman, 2011, S. 922). Die Autoren beschreiben das Experiment als eine Forschungsmethode, die den Zweck hat, unter der Nutzung von abhängigen und unabhängigen Variablen, Hypothesen zu testen.
Insbesondere die Planung ist ein wichtiger Bestandteil des naturwissenschaftlichen Experimentierens. Beim Tinkering hingegen muss das Ziel nicht von Beginn an feststehen. Es wird ein allgemein gehaltenes Ziel formuliert, das ständig adaptiert wird. Die Fähigkeit des Tinkerings wird als sehr wichtig für die heutige Gesellschaft gesehen. In der heutigen Zeit ist es wichtig, innovative Lösungen für teilweise unerwartete Probleme zu finden, da Gegebenheiten rapiden Veränderungen unterliegen. Das naturwissenschaftliche Experiment ist hingegen eine bewährte Arbeitsweise, um wissenschaftliche Erkenntnisse zu gewinnen (vgl. Kapitel 2.2).
Darüber hinaus beschäftigen sich Resnick und Rosenbaum (2013) mit der Frage, wie Technologien entwickelt werden müssen, um für das Tinkering geeignet zu sein. Sie formulieren drei Prinzipien, denen sie bei der Entwicklung von Geräten gefolgt sind: direktes Feedback,

flüssiges Experimentieren und offenes Entdecken.
Resnick und Rosenbaum (2013, S. 166) motivieren die Wichtigkeit von Tinkering wie folgt: „Schools tend to emphasize the value of planning, teaching students to analyze all options, develop a strategy, then carry out the plans. That's why students who are natural planners tend to do well in school. But what about students who are natural tinkerers? They often feel left out or alienated, especially in science, technology, engineering, and math (STEM) classes, which particularly emphasize top-down planning. Thus, many students are turned off to math and science, leading to a less scientifically literate populace and a restricted pipeline to STEM professions". Dieses Zitat verdeutlicht den Ausgangspunkt des Tinkering, der sich bewusst von dem geplanten Vorgehen distanziert. Da jedoch ein Top-Down-Vorgehen im STEM-Bereich kultiviert wird, beschäftigt sich diese Dissertation mit dem strukturierten Vorgehen in Physical-Computing-Aktivitäten. Systematische Untersuchung und detaillierte Phasenbeschreibungen in diesem Bereich sind bislang rar oder beziehen sich nur auf Teilaspekte des Physical-Computing-Prozesses.
Im Folgenden werden Ausschnitte aus der existierenden Literatur vorgestellt, die bereits auf den Prozess des Physical Computing hindeuten, ohne diesen konkret zu untersuchen. O'Sullivan und Igoe (2004, S. xix - xxix, 49 - 86) beschreiben implizit einen Prozess, da sie erklären wie mit den Geräten umgegangen werden sollte und wie diese konstruiert werden. Als Anfangspunkt wird eine Zielvorstellung formuliert, bei der beschrieben wird, was eine Person im Umgang mit dem Gerät sehen, fühlen und/oder hören soll. Dies gilt für alle Personen, unabhängig davon, ob sie das Gerät selbst konstruieren oder ausschließlich mit dem fertigen Artefakt interagieren. In dieser Phase der Vorbereitung auf ein konkretes Produkt soll die interagierende Person in den Vordergrund rücken und die Anforderungen an das Objekt natürlichsprachlich formulieren, ohne auf die konkrete Programmierung einzugehen. Das heißt, es soll ein Ansatz gefunden werden, um ein formuliertes Problem zu lösen. Hier beschreiben die Autoren bereits, dass dieser Prozess in einer Schleife abläuft und die Anforderungen immer wieder neu formuliert und adaptiert werden müssen. Abhängig von der Vorerfahrung der durchführenden Person sollten möglichst am Anfang Probleme in späteren Phasen antizipiert werden. Um diese zu bewältigen, können verschiedene Plattformen für sogenannte *Maker* genutzt werden. Darauf werden Probleme geteilt und online nach Lösungen gesucht. Bei diesem Gedanken wird davon ausgegangen, dass es sehr wahrscheinlich ist, dass dieses oder ein ähnliches Problem schon länger existiert und bei weiteren Makern aufgetreten ist. Eine allgemeine Empfehlung befindet sich ebenfalls in der Beschreibung der Vorbereitung. Die formulierte Aufgabe sollte in mehrere Teile zergliedert werden, so dass es möglich ist, an anderen Teilen und Problemen weiterzuarbeiten, falls die Person mit einem Problem nicht vorankommt. Somit kann weiterhin an dem Projekt gearbeitet werden und die Lösung eines akuten Problems ist nicht (zwangsläufig) umgehend erforderlich.
Nachdem die allgemeine Vorbereitung abgeschlossen ist, beschreiben die Autoren eine Implementationsphase, die sich zunächst auf technische Aspekte bezieht. Es müssen kon-

krete Materialentscheidungen getroffen werden, wie z. B., ob es sich um einen analogen oder digitalen Input handelt sowie um Verarbeitungsgeräte und einen Output. Wichtige Entscheidungen können an dieser Stelle ebenfalls sein, ob Prozesse parallel oder seriell ablaufen sollen. Ist die technische Konstruktion geplant, wird diese anschließend praktisch umgesetzt und ein entsprechender Programmcode implementiert. Es wird beschrieben, dass Sensordaten ausgelesen werden müssen, Aktuatoren an- und ausgeschaltet werden und eine Kommunikation mehrerer Geräte untereinander, sowie das damit verbundene Senden und Empfangen von Daten, eine Rolle spielen kann.
Anschließend muss das Programm ausgeführt werden, um es in der Praxis zu testen. Hier wird darauf hingewiesen, dass für jeden Testdurchlauf lediglich eine Variable verändert werden sollte, um das Programm angemessen zu testen.
Beim Testen bzw. Ausführen des Programms muss beobachtet werden, welche Auswirkungen nicht nach der Zielvorstellung ablaufen, um das zugrundeliegende Problem schlussfolgern zu können. Es wird davon ausgegangen, dass mehrere Durchläufe nötig sind und es lediglich in der Theorie möglich ist, beim ersten Testlauf ein vollständig funktionsfähiges System zu erzeugen. Als Möglichkeit der Fehlerbehebung beschreiben O'Sullivan und Igoe das strukturierte Ersetzen von Hardware- und Softwarebestandteilen zur Identifikation der Problemursache.
Banzi beschreibt ebenfalls Teile des Physical-Computing-Prozesses. Den Umgang mit Arduinos bezeichnet er als *Interaction Design*, wobei es sich dabei um eine Art Design handelt, welches das prototypische Arbeiten in den Mittelpunkt der Methode rückt (Banzi, 2011, S. 5). Eine klare zielorientierte Perspektive zeigt sich in der Aussage „classic engineering relies on a strict process for getting from A to B; the Arduino Way delights in the possibility of getting lost on the way and finding C instead". Diese Beschreibung lässt auf einen weniger geplanten Prozess schließen. Es wird ein klares Ziel verfolgt, jedoch soll stets Raum für Veränderungen und Abweichungen gewahrt werden.
Okita hingegen näherte sich dem Physical Computing eher auf einer prozessorientierten Ebene im Bereich der Problemlösung mit Robotern. Sie beschreibt in ihrer Untersuchung das sogenannte *rekursive Feedback* (Okita, 2014, S. 15), das wie folgt definiert wird: „In robotics programming, there is a recursive loop where programmers map their understanding to what they observe in their robot's behavior, and any discrepancies they notice maps back to the formalized rules they used for programming." Die ProgrammiererInnen können das Programm zunächst schreiben und nach der Ausführung anpassen, jedoch existiert dazwischen für die Personen keine Möglichkeit Veränderungen an dem System vorzunehmen. Aus dem Wissen des programmierten Inputs und den Auswirkungen muss nachgelagert auf die Diskrepanz geschlossen werden.
Einige ForscherInnen beschreiben den Physical-Computing-Prozess als eine Art des Experimentierens. Teilweise wird das Ziel und die Vorgehensweise dafür nicht klar definiert. Diese Beschreibung bezieht sich auf Beobachtungen in verschiedenen Studien oder auf normative Festlegungen. Inwieweit diese Daten einer soliden empirischen Basis entstammen,

wird in einigen Quellen nicht deutlich. Die hier aufgezeigten Beschreibungsansätze für den Physical-Computing-Prozess wurden bislang nicht zu einem detaillierten Gesamtprozess synthetisiert. Des Weiteren beziehen sich die meisten Beschreibungen lediglich auf den Umgang mit einem konkreten Physical-Computing-Gerät, was eine Generalisierung auf den Gesamtprozess erschwert.

2.1.1 Forschungsergebnisse in interdisziplinären Physical-Computing-Projekten

Physical Computing beschränkt sich nicht auf eine Anwendung in der Informatik, sondern erstreckt sich insbesondere über die MINT-Fächer. Auch in interdisziplinären Kontexten ist es bereits sehr populär, was der Spektrum an Forschungsergebnissen erweitert.
Das *Arts & Bots*-Projekt der Carnegie Mellon University hat sich zum Ziel gesetzt, Robotik sowie Physical Computing allgemein mit dem Basteln verschiedener Materialien zu verbinden. Dafür sollen Lernende mit verschiedenen Fähigkeiten und Interessen mit Informatik und Ingenieurswissenschaften in Kontakt gebracht werden. Physical-Computing-Geräte werden jeweils mit einem weiteren nicht-technischen Fach verbunden. In diesem Fall wurden die Fächer Englisch, Geschichte und Naturwissenschaften gewählt (Hamner et al., 2016) und ein Fortbildungsprogramm für Lehrkräfte entwickelt, um sie zu befähigen diesen Unterricht durchzuführen. Die Lehrkräfte sollen dabei nötige Fähigkeiten erwerben und ihr Selbstvertrauen stärken, um diese Inhalte in ihren Unterricht zu integrieren. Dieses Konzept adressiert SuS in der Mittelstufe in amerikanischen Schulen. Ein Beispiel für den Englischunterricht ist das Vortragen eines Gedichtes, welches durch Physical Computing unterstützt wird. Das kann z. B. durch verschiedenfarbige blinkende LEDs realisiert werden, wobei die Farbe der LEDs andeutet, welches Tier aktuell in dem Gedicht oder einer Geschichte auftaucht.
In einer weiteren Publikation wird die Integration des Konzepts in Schulklassen der Sekundarstufe evaluiert (Cross et al., 2015). Als ein konkretes Ziel wird benannt, mehr Diversität in die Sekundarschule zu bringen und den Status quo aufzubrechen. Insbesondere Frauen sind in STEM/MINT-Fächern unterrepräsentiert. Das verwendete genderneutrale Robotik-Kit und die Materialien zum Basteln sollen dabei Kreativität und Kunst miteinander verbinden. Neben dem zuvor angesprochenen Beispiel der Darstellung von Gedichtsausschnitten, wurden bewegliche historische Figuren nachgebildet. Ein dritter Bestandteil ist das muskuläre System des Körpers. Dieses wurde mit der Gesundheitslehre und dem Sportunterricht verbunden, wobei das Muskelsystem und die Gelenke des Menschen nachgebildet wurden. Evaluiert wurde das Projekt mittels Pre- und Posterhebung durch Fragebögen. Ein Fragebogen wurde zur Bestimmung des Lernzuwachses im Bereich Hardware, Software und Systemengineering eingesetzt. Ein weiterer Fragebogen (bzw. Selbstreport) wurde verwendet, um die Einstellung der Lernenden über die Subskalen zu erheben: Motivation, Interesse, Neugierde und Zuversicht/Identität.

Ergebnisse der Studie zeigen einen Lernzuwachs bei den Lernenden. In der Subskala des Systemengineering-Verständnisses, zeigt sich eine signifikante Verbesserung vom Pre- zum Posttest ($N = 138$). Der Fragebogen, der die Einstellung zum technischen Lernen erhob, zeigte ebenfalls signifikante Unterschiede. Es handelt sich jedoch um einen Abstieg der Zustimmung zur Aussage, dass die Lernenden gespannt sind wie Physical Computing funktioniert und gerne mehr darüber erfahren möchten. Anhand des Selbstreports und des Fragebogens zum Lernzuwachs konnte ebenfalls ein Zuwachs an Selbstvertrauen beim Umgang mit Technik festgestellt werden. Die Lernenden gaben außerdem an, dass sie ein besseres Verständnis für die Programmierung erworben haben und ihre Teamfähigkeit verbesserten. Aufgrund der diversen Stichprobe kommen Cross et al. zu dem Schluss, dass das Projekt *Arts & Bots* Menschen verschiedener demographischer Gruppen, Geschlechter und technischen Vorwissens verbindet.

Yannier et al. kommen zu dem Ergebnis, dass eine Kombination von physikalischen Experimenten und einer Umgebung zum Anfassen (Feedback durch Computerspiele), einen substantiellen Einfluss auf das Lernen und das Engagement für Naturwissenschaften hat (Yannier et al., 2013). Neben der Verbesserung in der Erläuterung physikalischer Prinzipien, zeigen die Ergebnisse auch produktive Kollaboration. Sie untersuchten Kinder im Alter von vier bis acht Jahren beim Umgang mit der interaktiven, anfassbaren Umgebung *EarthShake*. Sie hatten in der Aufgabe Vorhersagen zu treffen, welches der Hochhäuser am ehesten auf dem sich bewegenden Tisch bei einem Erdbeben falle. Ziel dieser Aufgabe ist es, den Kindern physikalische Prinzipien von Stabilität und Balance beizubringen.

In der deutschsprachigen Literatur existieren einige Unterrichtsansätze zum fächerverbindenden MINT-Unterricht mittels Physical Computing. So bieten zum Beispiel Händler wie LEGO Unterrichtskonzepte zum Einsatz ihrer Produkte an (The LEGO Group, 2016a,b). Darüber hinaus gibt es einzelne wissenschaftliche oder praktische Beiträge, die ein ausgewähltes Thema behandeln (Kocanda et al., 2010). Oftmals fehlt jedoch der Bezug zum Rahmenlehrplan der verbundenen Fächer. Es sollte klar herausgestellt werden, welche Inhalte und Kompetenzen in den einzelnen Fächern adressiert werden, da dies die Integration fächerverbindender Unterrichtseinheiten rechtfertigen und die Motivation der Lehrkräfte für neue Zugänge erhöhen kann.

Um diese Lücke zu adressieren, wurde eine systematische Analyse der Rahmenlehrpläne in den MINT-Fächern vorgenommen und mögliche Überschneidungen aufgezeigt (Lindner et al., 2017). Aufbauend darauf werden konkrete Unterrichtseinheiten vorgestellt, die Inhalte und Kompetenzen der Informatik mit denen aus jeweils einem anderen MINT-Fach verbinden. Auf der theoretischen Ebene der Rahmenlehrpläne erweist sich Physical Computing als geeigneter Kandidat für fächerverbindenden Unterricht. In der praktischen Durchführung beim Girls'Day 2017 nahmen sieben Schülerinnen der 9. Klasse an einem

Unterrichtsbeispiel aus dem Bereich Biologie und Informatik teil. Sie gaben an, dass sie ähnliche Inhalte mit Robotern gerne im Schulunterricht behandeln würden.
Ein weiterer Schwachpunkt an existierenden Konzepten ist, dass es oftmals an einer systematischen Evaluierung mangelt. Eine Pilotierung zum Einsatz von fächerverbindenden Inhalten mit LEGO Mindstorms-Robotern wurde von Schulz und Pinkwart mit einer Befragung von angehenden Lehrkräften aus den MINT-Wissenschaften durchgeführt (Schulz und Pinkwart, 2015). Dabei führten sie einen Workshop mit den Studierenden durch, bei dem sie interdisziplinäre Aufgaben aus der Biologie und Physik mit LEGO Mindstorms-Robotern bearbeiteten. Anschließend sollten sie bewerten, wie angemessen diese Aufgaben sind, um Kompetenzen in der Informatik, Biologie und Physik zu erwerben. Diese Kompetenzen wurden internationalen Standards entnommen (Gott et al., 2003; Next Generation Sciene Standards, 2017; Seehorn et al., 2011). Die Ergebnisse deuten ebenfalls darauf hin (Stichprobengröße $N = 9$), dass Physical Computing ein geeigneter Kandidat ist, um fächerverbindenden MINT-Unterricht in Schulen durchzuführen. In internationalen Standards geforderte Kompetenzbereiche wurden wiedererkannt und zumeist als „einfach in den Unterricht zu integrieren“ bewertet.

2.2 Stand der Forschung in der wissenschaftlichen Erkenntnisgewinnung der Naturwissenschaften

Aus Kapitel 2.1 wird bereits deutlich, dass kaum Modelle zur Beschreibung eines Physical-Computing-Prozesses existieren, wobei insbesondere die interdisziplinären Forschungsprojekte auf Verknüpfung von MINT/STEM-Inhalten untereinander Bezug nehmen. Die aus der Literatur extrahierten Bestandteile des Physical-Computing-Prozesses (vgl. Resnick und Rosenbaum, 2013; Banzi, 2011; Okita, 2014; O'Sullivan und Igoe, 2004) scheinen jedoch viele Gemeinsamkeiten mit der wissenschaftlichen Erkenntnisgewinnung zu haben. Insbesondere durch herausgestellte inhaltliche Überschneidungen innerhalb von MINT/STEM stellt sich die Frage, ob SuS und WissenschaftlerInnen ebenfalls einen ähnlichen Prozess beim Lösen von Problemen durchlaufen. Um diese Beobachtung untersuchen zu können, wird im Folgenden die Verortung und der Forschungsstand der wissenschaftlichen Erkenntnisgewinnung eingeführt.
Bei den Naturwissenschaften handelt es sich um ein breites wie tiefes Forschungsgebiet, dem eine lange Geschichte zu Grunde liegt. Auch die didaktische Forschung kann auf eine große Vielfalt von Erkenntnissen zurückgreifen. Ein großes Forschungsfeld ist die naturwissenschaftliche Erkenntnisgewinnung, welche in die Theorie des Problemlösens einzuordnen ist (Mayer, 2007; Klahr, 2000). International wird die (natur-)wissenschaftliche Erkenntnisgewinnung als Scientific Inquiry (SI) oder Nature-of-Science (NoS) bezeichnet. In Deutschland sind auch die Begriffe wissenschaftliche Denk- und Arbeitsweisen gängig. Inhaltlich gibt es marginale Unterschiede in der Verwendung dieser Begrifflichkeiten, auf

die im Folgenden kurz eingegangen wird. Arndt vergleicht in ihrer Dissertation verschiedene Modelle der wissenschaftlichen Erkenntnisgewinnung und greift die herausgestellten Unterschiede in nationaler und US-amerikanischer Literatur auf (Arndt, 2016). In dem Vergleich kommt Arndt zu dem Schluss, dass die inhaltliche Bedeutung der *wissenschaftlichen Erkenntnisgewinnung* als äquivalent zum *Scientific Inquiry* betrachtet werden kann. In dieser Arbeit wird es analog behandelt. Die Relevanz dieses Forschungsfeldes wird durch ihre Verankerung in verschiedenen nationalen (Kultusministerkonferenz, 2005) und internationalen Standards (National Research Council, 1996) gestützt. Die wissenschaftliche Erkenntnisgewinnung wird als wichtige Komponente der naturwissenschaftlichen Grundbildung deklariert (American Association for the Advancement of Science and others, 1993).

Mayer setzt die Standards der Erkenntnisgewinnung in Beziehung zu beschriebenen Kompetenzkonstrukten aus der Kognitionspsychologie. In seinem Modell (siehe Abbildung 2.1) sind in Klammern jeweils die internationalen bzw. US-amerikanischen Bezeichnungen zu finden. Des Weiteren gliedert er die Erkenntnisgewinnung in die Bereiche *Charakteristika der Naturwissenschaften (Nature-of-Science)*, *wissenschaftliche Untersuchungen (Scientific Inquiry)* und *wissenschaftliche Arbeitstechniken (Practical Work)* auf. Dem Bereich *Charakteristika der Naturwissenschaften* wird das *Wissenschaftsverständnis* als Kompetenzstruktur zugeordnet. Zu den Charakteristika gehört eine Vorstellung darüber, wie WissenschaftlerInnen forschen und ihre Erkenntnisse gewinnen (Höttecke, 2001). Die Dimension *wissenschaftliche Arbeitstechniken* ist eng an manuelle Fertigkeiten geknüpft, beispielsweise durch praktische Durchführungen wie physikalische Messungen oder dem Mikroskopieren. Im Zentrum von Mayers Beitrag steht jedoch die Kompetenz des *wissenschaftlichen Denkens*, die dem Standard der wissenschaftlichen Untersuchungen zuzuordnen ist. Mayer stellt in diesem Beitrag auch die begrifflichen Äquivalenzen her. International wird wissenschaftliches Denken in den verschiedenen Forschungsbereichen auch als Scientific Reasoning, Procedural Understanding, Concepts-of-Evidence, Process Skills oder Inquiry Skills bezeichnet. Das wissenschaftliche Denken wird stark von Wissenschaftsvorstellungen beeinflusst. Ebenfalls sind die manuellen Fähigkeiten notwendig, um wissenschaftliche Schlüsse aus Untersuchungen zu ziehen. Damit bedingen sich die Kompetenzkonstrukte untereinander und stehen in Beziehung zu den Standards der Erkenntnisgewinnung.

Neben den im Modell enthaltenen Aspekten, stellt Bybee eine weitere Dimension des Scientific Inquiry heraus (Bybee, 2002, S. 28 - 29). Diese adressiert den konzeptionellen Wandel der Lernenden über Scientific Inquiry als Lernmethode. Für diesen Wandel gilt, dass er zu einem tieferen Verständnis führen soll. Nach Bybee ist das nur durch eine feste Implementation in den Unterricht realisierbar: „Clearly, the contemporary view of how students learn implies content that is deeper than facts and information, a curriculum that is richer than reading, instruction that is longer than a lesson, and teaching that is more than telling.“

Klahr und Dunbar (1988) teilen den Bereich der wissenschaftlichen Untersuchungen in

den Hypothesen- und Experimentierraum in dem sogenannten *SDDS-Modell* (Scientific Discovery as Dual Search). In dem ersten Teil wird das Problemfeld des Hypothesenraums untersucht und eine zu testende Hypothese aufgestellt. Anschließend teilt sich der Experimentierraum in das Testen der Hypothese und die Evaluation der erhobenen Daten. Dieser Teil inkludiert unter anderem die Entwicklung eines Versuchsaufbaus und die Beobachtung von Ereignissen. Es gibt verschiedene Modelle, die den Prozess der wissen-

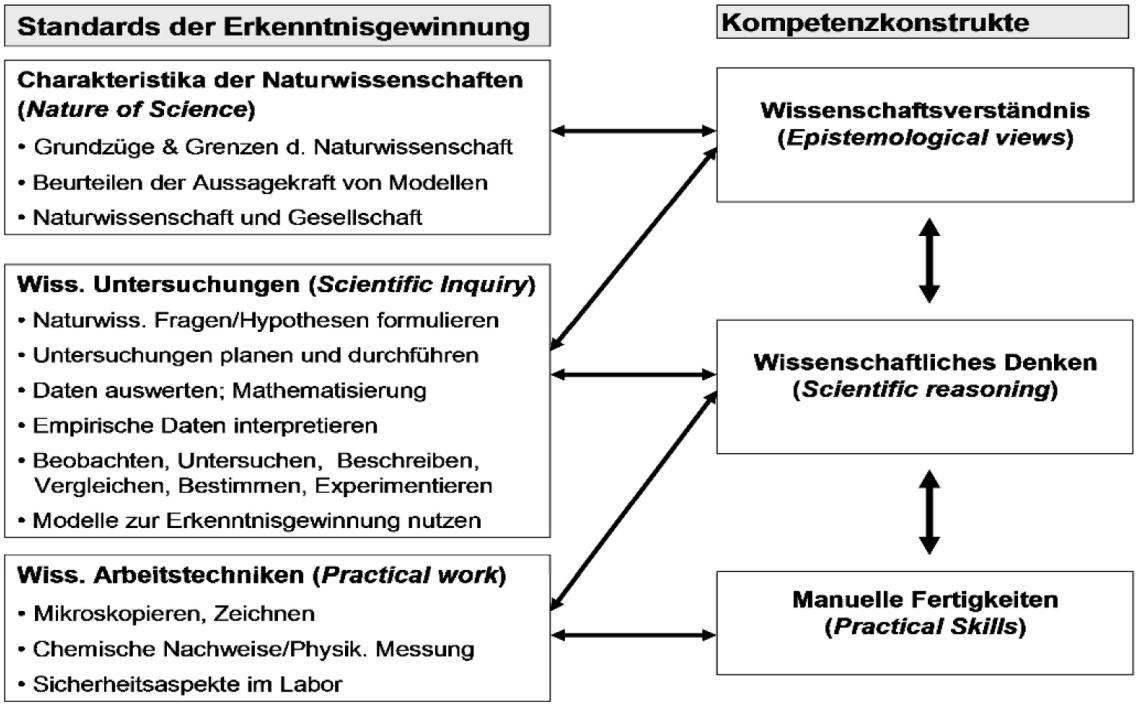


Abbildung 2.1: Rahmenkonzept wissenschafts-methodischer Kompetenzen nach Mayer (2007)

schaftlichen Erkenntnisgewinnung beschreiben und zumeist sind die verschiedenen Denk- und Arbeitsweisen ähnlich. Die Denkweisen, die zugleich den Prozess des Scientific Inquiry darstellen, können in Beobachtung, Hypothese, Interferenz, Test und Feedback gegliedert werden (Bybee, 2002). Als Arbeitsweisen werden Beobachtung, Experimentieren (Bybee, 2002) und das Modellieren (White und Frederiksen, 1998; Upmeier zu Belzen und Krüger, 2010) genannt.

Bei dem Experimentierprozess wird im Folgenden ein Modell aus der Didaktik der Physik herangezogen, um die ablaufenden Teilprozesse zu beschreiben (siehe Abbildung 2.2). Es dient ebenfalls in einem folgenden Kapitel als Exempel zum Vergleich von naturwissenschaftlichen Prozessen zur Erkenntnisgewinnung und denen der Informatik (vgl. Kapitel 3.1). In diesem Modell wird der Prozess in drei Hauptphasen gegliedert: Planung, Durchführung und Auswertung. Diesen werden Unterphasen zugeordnet, wobei auch Unterphasen aufgezeigt werden, die nicht konkret zu einer Phase zuzuordnen sind. In der *Planungsphase* soll zunächst die vorgegebene und zu untersuchende Fragestel-

lung geklärt bzw. eine eigene entwickelt werden. Davon abgeleitet werden Erwartungen formuliert und eine Hypothese über den Ausgang der Untersuchung gebildet. Im Übergang zur *Durchführungsphase* wird ein Versuchsplan entworfen. Eindeutig der Phase der Durchführung zugeordnet sind die Geräteauswahl und der Aufbau der Versuchsanordnung, um anschließend die geplanten Messungen durchzuführen und zu dokumentieren. Einen Übergang zur nächsten Phase bildet hier der Umgang mit Problemen und Fehlern, welcher sowohl in der Durchführung als auch *Auswertung* eine wichtige Rolle einnimmt. In der Phase der Auswertung werden die zuvor aufgenommenen Messdaten aufbereitet und verarbeitet, um anschließend ein Ergebnis interpretieren zu können. Dieses Ergebnis wird nun auf die anfänglich aufgestellte Hypothese und Fragestellung zurück bezogen, um sie beantworten zu können. In dem Modell gibt es keine feste Reihenfolge der Unterphasen, sondern es wird viel mehr darauf hingewiesen, welche Phasen im Allgemeinen zu beobachten sind. Es kann also eine Verkürzung oder Abwesenheit von Unterphasen stattfinden. Schreiber et al. (2009) beschreiben ebenfalls die Iteration von Teilen des Prozesses bis hin zum vollständigen Prozess.

Für die konkrete Unterteilung der Arbeits- und Denkweisen werden verschiedene Be-

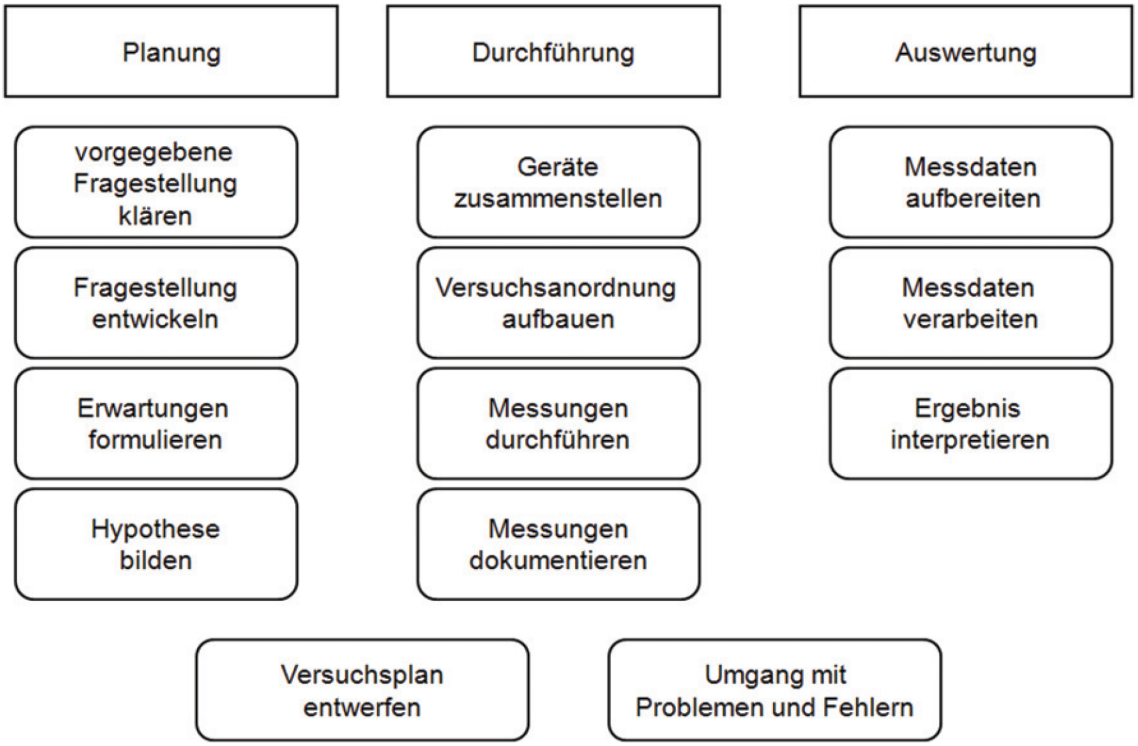


Abbildung 2.2: Modell experimenteller Kompetenz nach Schreiber et al. (2009)

zeichnungen und Kategorisierungen verwendet, wobei überwiegend ähnliche Phänomene beschrieben werden.

In der naturwissenschafts-didaktischen Forschung wird auf verschiedene Phasen des Pro-

zesses fokussiert. Abhängig von der konkreten Fachrichtung werden die Unterkategorien bzw. Phasen unterschiedlich beschrieben. In dem Beitrag von Schreiber et al. (2009) fokussieren sich die AutorInnen vor allem auf die Beschreibung von Kompetenzen in der Phase der Durchführung, Mayer (2007) hingegen auf die Phasen der Hypothesenbildung, der Planung von Untersuchungen und die Evaluation von Daten, wobei die Durchführungsphase explizit ausgeschlossen wird. Klahr und Dunbar konzentrieren sich ebenfalls weniger auf die Durchführungsphase, sondern betrachten die Planungsprozesse innerhalb der Durchführung (vgl. Arndt, 2016; Schreiber et al., 2009).
In einer Untersuchung validierten Patzwaldt et al. die Phasen und Teilphasen des Erkenntnisgewinnungsprozesses in der Chemie hinsichtlich der benötigten Experimentierkompetenzen. Dabei konnten sie fast alle in der Literatur beschriebenen Phasen identifizieren und ebenfalls beobachten, dass die Phase der Auswertung (oder auch als Evaluation bezeichnet) teilweise verkürzt oder übersprungen wird. In einer Studie teilten sie die zu lösende Aufgabe in Teilaufgaben und konnten daraus folgende Ergebnisse gewinnen (Patzwaldt et al., 2014, S. 185):

1. “die kognitiven Prozesse verlaufen parallel zur jeweils bearbeiteten Teilaufgabe (linear)“,
2. „der Dreischritt Planung – Durchführung – Auswertung wiederholt sich (iterativ)“,
3. „während der Bearbeitung der Teilaufgabe zur Durchführung sind mehrere Planungsphasen beobachtbar (oszillierend)“.

Die oben beschriebene Literatur zur wissenschaftlichen Erkenntnisgewinnung ist lediglich ein Ausschnitt aus diesem Forschungsfeld. Sie dient der inhaltlichen Beschreibung der Phasen und des Phasenverlaufs, wobei auf gängige Modelle in der Naturwissenschafts-Didaktik zurückgegriffen wurde. Die Modelle geben Lehrkräften einen Überblick, wie der Experimentierprozess ablaufen kann bzw. gestaltet werden sollte. Ebenfalls können sie als Hilfestellung genutzt werden, um SuS durch den Prozess zu leiten und die kognitive Beanspruchung zu reduzieren (vgl. Sweller et al., 1998). Der gewählte Ausschnitt ermöglicht einen Einstieg in die Thematik der wissenschaftlichen Erkenntnisgewinnung. Darüber hinaus wird die Grundlage gelegt, um den Physical-Computing-Prozess mit dem Prozess der wissenschaftlichen Erkenntnisgewinnung zu vergleichen. Mögliche Ähnlichkeiten wurden bereits im Rahmen der Physical-Computing-Forschung erwähnt. Aus diesem Grund werden in den folgenden Abschnitten weitere Facetten der wissenschaftlichen Erkenntnisgewinnung vorgestellt.

2.2.1 Identifizierte Probleme während des Prozesses der wissenschaftlichen Erkenntnisgewinnung

Neben den benötigten Fertigkeiten, die Schülerinnen und Schüler mitbringen/erwerben müssen, um wissenschaftliche Erkenntnisse zu erlangen, soll in diesem Abschnitt auf

häufige Probleme der SuS eingegangen werden. Diesbezüglich wurden bereits die Phasen der wissenschaftlichen Erkenntnisgewinnung untersucht. De Jong, Van Joolingen und Zimmerman nahmen auf diesem Gebiet eine umfangreiche Literaturrecherche vor (De Jong, 2006; De Jong und Van Joolingen, 1998; Zimmerman, 2007). Dabei stellten sie fest, dass Lernende in allen Phasen der naturwissenschaftlichen Erkenntnisgewinnung diverse Probleme haben.
Viele Probleme sind bereits in der Phase der Planung und Hypothesenbildung zu verorten. Zumeist bevor die SuS überhaupt andere Phasen innerhalb des Experiments oder der Untersuchung begonnen haben. Dazu gehört, dass es ihnen Schwierigkeiten bereitet Hypothesen aufzustellen, die getestet werden können. SuS haben insbesondere Probleme, ihre experimentellen Daten auf ihre Hypothese zu beziehen, falls die Daten der Hypothese widersprechen. Sie scheinen häufig an ihren früheren Ideen festzuhalten, weshalb sieben verschiedene Reaktionen auf die widersprechenden oder anomalen Daten beobachtet werden konnten. Chinn und Brewer beschreiben diese Verhaltensweisen der SuS wie folgt. Sie:

1. ignorieren die anomalen Daten, d. h. sie erklären Daten nicht und erhalten ihre ursprüngliche Hypothese unverändert davon aufrecht. Es wird von den Autoren als der extremste Weg, Daten abzufertigen, beschrieben.

2. weisen die Daten zurück. Im Gegensatz zum Ignorieren von Daten wird gegebenenfalls geäußert, warum die Daten zurückgewiesen werden und keinen Einfluss auf die Hypothese haben.

3. schließen die Daten aus. Dabei bewerten die SuS, dass die Daten nicht zugehörig zur Hypothese A sind und sie damit nicht beeinflussen.

4. halten die Daten zunächst in der Schwebe. Insbesondere, wenn keine unmittelbare Vermutung für die Interpretation der Daten besteht, ist es möglich, dass SuS weiter eine Hypothese untersuchen, wobei sie sich mit den vorliegenden Daten später beschäftigen wollen. Es führt ebenfalls dazu, dass die Hypothese unverändert bleibt.

5. reinterpretieren die Daten. Die SuS akzeptieren die Existenz der Daten und setzen sich mit ihnen auseinander, wobei sie versuchen die Daten mit ihrer Hypothese zu erklären und halten dabei an dieser fest.

6. nehmen leichte Veränderungen an Hypothese A vor. In diesem Fall akzeptieren sie ebenfalls die Daten und nehmen kleine Veränderungen an der Hypothese vor, was sie an Hypothese A festhalten lässt. Sie sind nicht gewillt, gänzlich auf eine Hypothese B zu wechseln. In dieser Stufe werden zum ersten Mal überhaupt Veränderungen vorgenommen.

7. verändern Hypothese A. Dabei handelt es sich um den stärksten Effekt von anomalen Daten. Dabei verändern die SuS eine oder mehrere theoretische Überzeugungen, die

sie zu ihrer ursprünglichen Hypothese veranlasst hatten. Sie akzeptieren die Daten und können diese durch die Veränderungen ihrer Überzeugung bezüglich Hypothese A erklären, oder sie akzeptieren eine alternative Hypothese B.

Es bereitet den SuS oftmals Probleme, aus den Ergebnissen richtige Schlüsse in Bezug auf ihre Hypothese zu ziehen (Chinn und Brewer, 1993).
Das Testen von Variablen bereitet ProbandInnen ebenfalls große Schwierigkeiten. Studierenden fällt es besonders schwer eine Hypothese in testbare Variablen zu überführen (Lawson, 2002). Wenn die vermutete Ursache nicht sichtbar ist und z. B. nur indirekt getestet werden kann, machten die Studierenden diverse Fehler in ihrer Argumentation. Lawson beschreibt drei fehlerhafte Argumentationstypen:

1. Argumentationen, in denen Elemente fehlten oder verworren sind,
2. Argumentationen, in der Prognosen getätigt werden, die nicht der Hypothese oder dem geplanten Test entsprechen,
3. Argumentationen, die alternative Hypothesen nicht abdecken.

Darüber hinaus untersuchte Keselman wie SuS mit mehreren, sich bedingenden Variablen umgehen. Ein Ergebnis war, dass z. B. SuS versuchen mehrere Variablen gleichzeitig zu verändern (Keselman, 2003). Probleme im Umgang und dem Testen von Variablen wurde von weiteren ForscherInnen beobachtet. Es wurde gezeigt, dass SuS bei der Entwicklung des Testdesigns zum Teil die falsche Variable zum Testen wählten, insbesondere wenn sie eine positive Behauptung gegeben hatten (Zimmerman und Glaser, 2001). Dabei ließen sich die SuS stark davon leiten, ob die zu testende Behauptung in eine positive oder negative Handlung eingebettet war.
SuS fällt es allgemein schwer sich dessen bewusst zu sein, wie ein Experiment durchgeführt wird. Dadurch kommt es oft dazu, dass sie nicht bemerken, wie sie nicht länger ihre Hypothese testen und versuchen einen bestimmten Zustand zu erreichen (Schauble et al., 1995). Das Finden von Mustern in Daten und dessen Interpretation stellte sich in dieser Studie als schwierig für die SuS heraus. Der Rückbezug der Daten auf die Hypothese und die Implikationen für Gesetzmäßigkeiten bereitet den SuS Probleme.
SuS haben außerdem Probleme dabei, ein Experiment zur Erkenntnisgewinnung einzusetzen. Es wird oftmals genutzt, um einen Effekt zu erzielen bzw. zur Bestätigung ihrer Hypothese (Wason, 1960). In der gleichen Publikation wird geschlussfolgert, dass SuS nicht in der Lage oder nicht gewillt sind, ihre eigenen Hypothesen zu testen. Schauble et al. (1991) beobachteten, dass vielen SuS nicht bewusst ist, dass auch scheinbar ungünstige Ergebnisse von Experimenten durchaus wertvolle Daten erzeugen können.
Die bis hier aufgeführte Literatur verdeutlicht, dass die wissenschaftliche Erkenntnisgewinnung ein wichtiges erklärtes Bildungsziel ist und ebenfalls eine große Herausforderung für SuS darstellt. Da im Physical Computing mit Sensoren gearbeitet wird, sind Messunsicherheiten nicht zu vermeiden und es kann davon ausgegangen werden, dass die SuS

ähnliche Probleme wie beim naturwissenschaftlichen Experimentieren haben werden. Die aufgezeigten Schwierigkeiten konnten bei SuS unterschiedlichen Alters beobachtet werden und scheinen nicht auf spezifische Altersstufen begrenzt zu sein. Abgeleitet von den Problemen wurden bereits verschiedene Hilfestellungen entwickelt, um konkrete Probleme zu adressieren. Eine Auswahl an Hilfestellungen werden im Folgenden aufgezeigt.

2.2.2 Untersuchungen zur Unterstützung des Prozesses der wissenschaftlichen Erkenntnisgewinnung

Im Bereich der wissenschaftlichen Erkenntnisgewinnung existieren bereits verschiedene Ansätze, wie dieser Prozess durch Technologien unterstützt werden kann. Dieser Abschnitt beschreibt existierende Tools und Methoden zur Unterstützung des Inquiry-Prozesses. Van Joolingen et al. (2007) geben dazu einen Überblick verschiedener computerbasierter Unterstützungsmöglichkeiten. Diese werden in vier Bereiche gegliedert:

1. Simulationen, die die Komplexität eines Sachverhalts verringern sowie gezielte Aspekte herausstellen können,
2. Tools, die den Inquiry-Prozess unterstützen, z. B. durch die Visualisierung von Daten. Diese Tools können teilweise spezifischen Phasen der wissenschaftlichen Erkenntnisgewinnung, die sie konkret unterstützen sollen, zugeordnet werden,
3. Kollaborationsunterstützungen, z. B. durch den Austausch von Daten,
4. Modellierungstools, die eine anschließende Simulation zulassen.

Die Unterstützung des Inquiry-Prozesses (vgl. Anstrich 2) wird durch ein intelligentes Tutorensystem in dem folgenden Beispiel skizziert. Da es SuS schwer fällt, testbare Hypothesen aufzustellen, wurde das *Scratchpad* entwickelt, welches eine Auswahl an Hypothesen anbietet. Die Lernenden haben die Aufgabe aus den bestehenden Hypothesen auszuwählen (van Joolingen und de Jong, 1993). Dabei zeigten die Autoren, dass ein Impuls zur Entwicklung einer Hypothese, bevor zum Experimentieren übergegangen wird, zu präziseren Hypothesen und einem besseren Resultat der Untersuchung führt.
Sandoval und Reiser entwickelten den *ExplanationConstructor* als unterstützendes Tool im Biologieunterricht (vgl. Anstrich 2). Dieses Tool kann der Unterstützung diverser Phasen des Inquiry-Prozesses zugeordnet werden, da es das Ziel der Konstruktion von Erklärungen eines Phänomens verfolgt. SuS werden durch konzeptuelle und erkenntnistheoretische Hilfestellungen unterstützt. Die visuelle Trennung von Beweisen und Behauptungen soll der Überversachlichung von Daten entgegenwirken, welche in der wissenschaftlichen Erkenntnisgewinnung ein viel beschriebenes Problem darstellt. Zur Untersuchung eines Phänomens ist das Programm als Tagebuch aufgebaut, worin die Fortschritte der Untersuchung dokumentiert werden. Es dient als epistemologisches Tool zur Unterstützung des Inquiry-Prozesses, indem Argumente formuliert und Modelle gebildet werden. Damit soll es vor

allem dazu beitragen, das Verständnis über Naturwissenschaften und deren Grenzen zu entwickeln (Sandoval und Reiser, 2004). Diese Studie untersuchte die generelle Einsetzbarkeit von Tools in diversen Lernszenarien. Sie kamen zu dem Ergebnis, dass erkenntnistheoretische Tools während wissenschaftlicher Untersuchungen bei der Strukturierung von Aussagen der SuS unterstützen können. Darüber hinaus schlussfolgern die Autoren, dass zusätzlich zu den prozessbezogenen Hilfestellungen ebenfalls soziale Unterstützungen für den Prozess gegeben werden sollten. Die Rolle von epistemischen Tools bewerten sie als einzigartig, um die Erkenntnisgewinnung von SuS zu unterstützen.
Es gilt zu überprüfen, inwieweit diese Strukturierung auf andere Fächer und Inhalte übertragbar ist. Prinzipiell scheint der Ansatz ebenfalls für den Informatikunterricht geeignet zu sein. Im Physical Computing werden Auswirkungen des Systems beobachtet und die Beobachtungen im Anschluss interpretiert (vgl. Okita, 2014). Eine visuelle Trennung von Daten sowie deren Dokumentation könnte dabei unterstützen. Es ist jedoch nicht zu vernachlässigen, dass dieses Vorgehen sehr zeitaufwendig ist. Deshalb sollte der konkrete Mehrwert untersucht werden.
Im Bereich des kollaborativen Problemlösens (vgl. Anstrich 3) wurde gezeigt, dass Instruktionen, die die Regulation durch diverse Guidelines unterstützen, einen positiven Einfluss auf die wissenschaftliche Erkenntnisgewinnung haben können (Manlove et al., 2006). Es konnte nachgewiesen werden, dass die SuS durch eine Unterstützung bei der Koordination des Prozesses die Planung ihrer Untersuchung verbesserten. Des Weiteren stieg die Qualität der entwickelten Modelle bei Gruppen, welche die Prozesskoordination nutzten.
Das kollaborative Problemlösen spielt im Physical Computing eine wichtige Rolle, da die SuS zumeist aus finanziellen Gründen in Gruppen arbeiten.

Die aufgezeigte Kategorisierung von Van Joolingen et al. (2007) bezieht sich ausschließlich auf computergestützte Hilfestellungen, die somit der heutigen Zeit angemessen erscheinen. Nichtsdestotrotz existieren ebenfalls weitgehend analog durchgeführte Untersuchungen, die zum Ziel haben den Inquiry-Prozess zu unterstützen.
Arnold et al. untersuchten in einer quantitativen Studie mit $N = 96$ SuS im Alter von 16 - 19 Jahren und in einer anschließenden qualitativen Analyse von zwei Paaren dieser Gruppe, welche Unterstützung SuS in der Erkenntnisgewinnung benötigen. Sie kamen zu dem Ergebnis, dass sie Unterstützung benötigen, um ein höheres Kompetenzlevel zu erreichen und in der Lage sind ihr Experiment zu reflektieren (Arnold et al., 2014). In der Untersuchung wurde eine offene Inquiry-Aufgabe gewählt und es wurde beobachtet, welche unterschiedlichen Hilfestellungen retrospektiv angebracht gewesen wären. Die hier getroffene Unterscheidung von Hilfen ist dichotom mit einerseits Hilfestellungen für prozedurales Wissen und andererseits Hilfe für prozedurales Verständnis. In den Bereich des prozeduralen Wissens fällt zum Beispiel die Situation, wenn SuS nicht wussten, wie sie an einer bestimmten Stelle weiterarbeiten sollten. Es zeigte sich eine Vielfalt an Problemen in diesem Bereich. Das prozedurale Verständnis zeichnet sich durch das Wissen über Konzepte

aus und das Wissen darüber, warum diese Konzepte wichtig sind, z. B. der Einfluss und die Auswirkungen von Störfaktoren innerhalb eines Experiments. Arnold et al. fassen zusammen, dass Hilfen auf verschiedenen Ebenen benötigt werden. Beispielsweise auf der Ebene von prozeduralem Wissen und prozeduralem Verständnis, aber auch abhängig von Kompetenzleveln, auf denen sich die SuS befinden, da sie meistens während der Argumentation von einem Level zum Nächsten gehen. Als geeignetes Mittel schlagen die AutorInnen einen Fundus an Karten vor, auf denen unterschiedliche Hilfen gegeben werden. Diese können sich die SuS selbstständig ansehen, sobald sie nicht weiter kommen. Die Karten können z. B. die Wiederholung von Definitionen bis hin zu exemplarischen Antworten umfassen. Es sollte sicher gestellt werden, dass die SuS wissen, was Hypothesen, abhängige und unabhängige Variablen, etc. sind. Zur Unterstützung des prozeduralen Verständnisses wird die Verwendung von *Concept Cartoons* vorgeschlagen (vgl. Keogh und Naylor, 1999).
Die Nutzung von analogen Hilfestellungen erscheint in einer Klassenraumsituation hilfreich, in der die SuS keinen festen Arbeitsplatz haben oder zwischen verschiedenen Orten wechseln müssen. Dies ist beispielsweise bei der Arbeit mit Robotern der Fall, wenn die Geräte Parcours absolvieren. Des Weiteren wäre eine Darstellung der Karten im Rahmen einer Applikation für einen Tablet-Computer denkbar. Die Herausforderung an dieser Stelle ist jedoch, wie die Hinweise für die SuS so gestaltet werden können, dass sie konkret genug sind, um ihnen Hilfestellungen zu geben. Werden die Hinweise zu konkret, kann auch die Gefahr bestehen, dass einige Hinweise nicht auf die Situation der SuS passen und sie gegebenenfalls in falschen Annahmen bestätigt werden.

Im Folgenden werden Ergebnisse vorgestellt, die diese Art der Hilfestellung behandeln. Verschiedene Vorgehensweisen bei der Strukturierung des Prozesses und der Form des Feedbacks stehen im Fokus der Untersuchungen. Anhand eines umfangreichen Literaturreviews zeigen Alfieri et al., dass entdeckendes Lernen ohne Hilfestellungen im Allgemeinen keinen Nutzen zeigt (Alfieri et al., 2011). Dabei stufen sie ab, dass direkte Instruktionen besser sind, als keine Unterstützung zu geben. Eine weitere Forschergruppe spricht sogar von „the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching“ (Kirschner et al., 2006, S. 75), da ihr Literaturreview zeigte, dass keine oder minimale Unterstützung von SuS weniger effektiv und weniger effizient als Instruktionen sind. Sie stellen ebenfalls einen starken Bezug zum Vorwissen der SuS heraus, da ein hohes Vorwissen zu einer *internen Anleitung* führen kann (Kirschner et al., 2006; Clark et al., 2012). Die Effektivität verschiedener Formen von Feedback während des Lernprozesses wurde unter anderem von McKendree (1990) untersucht. Studierenden wurden verschiedene Formen von Feedback gegeben, wobei sich herausstellte, dass die Form einen Einfluss auf die Effektivität des Feedbacks haben kann. Dabei ist die Benachrichtigung, dass ein Fehler vorliegt, weniger effektiv zum Lernen und Problem lösen, als eine Erläuterung über die Art des Fehlers und Möglichkeiten zur Fehlerbehebung. Dadurch leiten sie ab, dass offene Instruktionen weniger förderlich sind als schrittweise Anweisungen und rein

informatives Feedback über einen Fehler jedoch in komplexeren und ambigen Aufgaben geeigneter sein könnte.
Es wurden bereits verschiedene Arten der Hilfestellung und deren Effektivität untersucht. In einer Studie mit 112 Dritt- und Viertklässlern wurde gezeigt, dass mehr SuS von direkten Instruktionen als von entdeckendem Lernen profitierten. Im Bereich von tiefer gehenden wissenschaftlichen Beurteilungen zeigte sich aber, dass SuS, die experimentelles Design durch direkte Instruktionen lernten, genauso gut abschnitten wie SuS, die die Methode selbstständig erarbeiteten (Klahr und Nigam, 2004). Zuvor wurde aufgezeigt, dass sich eine Kombination von direkten Instruktionen mit Untersuchungen als effektiv erweist, um wissenschaftliches Denken zu fördern (Chen und Klahr, 1999). Ebenso haben sich Unterstützungen beim entdeckenden Lernen mittels Simulationen als effektiv erwiesen (White und Frederiksen, 1998; Ketelhut et al., 2010). Bei dieser Fülle an Unterstützungsansätzen weisen die Autoren jedoch auch darauf hin, dass es zwei verschiedene Effekte gibt, die aus der Unterstützung resultieren sollten. Zunächst vordergründig ist, dass die SuS den Inquiry-Prozess besser absolvieren. Anschließend soll jedoch ebenfalls ein Lernzuwachs erreicht werden, auch wenn die Unterstützung nicht mehr zur Verfügung steht (Van Joolingen et al., 2007).

Zusammenfassend kann auf vielfältige Forschungsergebnisse im Bereich von Unterstützungssystemen zur wissenschaftlichen Erkenntnisgewinnung aufgebaut werden. Es liegen Ergebnisse vor, dass direkte Instruktionen besonders effektiv als Hilfestellung in der wissenschaftlichen Erkenntnisgewinnung sind. Aufgrund der Beschreibung von Physical Computing als eine Form des Experimentierens (vgl. Resnick und Rosenbaum, 2013) eröffnet sich ein neues Forschungsfeld, das Hilfestellungen für Physical-Computing-Aktivitäten adressiert. Es liegt nahe, dass die Notwendigkeit von Unterstützungsmöglichkeiten ebenfalls für den Bereich Informatik existiert.
Eine Kombination mit entdeckendem Lernen wird jedoch ebenfalls als geeignet beschrieben. Gegenstand der Forschung ist insbesondere die Unterstützung des Problemlöseprozesses sowie der Kollaboration von SuS. Designvorschläge zur Unterstützung einzelner Phasen und bewährte Vorgehensweisen im Problemlösen wurden aufgezeigt. Bislang wurden diese Tools vorrangig für den mathematisch-naturwissenschaftlichen Unterricht entwickelt und die Anwendbarkeit dieser Tools auf das Problemlösen in der Informatik wird bislang vernachlässigt. Die Verbindung von computergestützten Tools und realen Experimenten ist zu diesem Zeitpunkt ebenfalls rar. Die Informatik scheint jedoch ein geeigneter Kandidat zu sein, um die Kommunikation von Hardware- und Softwarekomponenten beim Experimentieren mit intelligenten Tutorensystemen zu verbinden und somit einen Mehrwert für den MINT-Bereich zu erschaffen.

2.2.3 Rolle vom kollaborativen Lernen

Wie bereits in Kapitel 2.2.2 erwähnt sind viele Lernszenarien im Physical Computing mit der Kollaboration von SuS verbunden. Ebenfalls bewirken einige entwickelte Unterstützungssysteme im MINT-Bereich gleichzeitig kollaboratives Lernen.
Darüber hinaus kann kollaboratives Lernen im Studiendesign gezielt genutzt werden, da durch die Diskussion in Gruppen Gedanken relativ ungefiltert geäußert werden, was Hinweise auf kognitive Prozesse ermöglicht. Um mögliche Einflüsse von Gruppenarbeit auf Studien auszuschließen bzw. abzuwägen, werden in diesem Abschnitt einige Forschungsergebnisse bezüglich des Einflusses von Diversität und Freundschaft betrachtet. Zur Abgrenzung aufkommender Terminologien ist zunächst zwischen kollaborativem Lernen (Collaborative Learning, CL) und kollaborativem Problemlösen (Collaborative-Problem-Solving, CPS) zu unterscheiden. Hesse et al. (2015) fassen *kollaboratives Lernen* als „jointly orchestrate their activities in order to address a particular task or problem“ (S. 38) zusammen. Bei dem gemeinsamen Bearbeiten von Aufgaben sind diese untrennbar miteinander verbunden, was das kollaborative Lernen vom kooperativem Lernen abgrenzt. Des Weiteren gehen die AutorInnen auf die dafür benötigten sozialen Fähigkeiten ein. Zum Tragen kommen ebenfalls die drei Elemente von Kollaboration: Kommunikation (zum Austausch der Personen), Kooperation (zur Arbeitsaufteilung) und Ansprechbarkeit (für eine aktive Partizipation). Als *Problemlösen* wird bezeichnet, wenn eine Diskrepanz zwischen einem aktuellen Zustand und einem Zielzustand herrscht, wobei es sich nicht um einen routinierten Lösungsweg handelt, um zum Zielzustand zu gelangen. Damit fassen Hesse et al. das *kollaborative Problemlösen* zusammen als eine gemeinsame Aktivität von kleinen Gruppen oder Paaren, die eine Anzahl von Schritten ausführen, um von einem aktuellen Zustand ihren Zielzustand zu erreichen. Dabei müssen sich die Gruppenmitglieder über das Problem austauschen, sowie über die Beziehung von Aktion und Auswirkung, und nach einer Lösung suchen. Diese Teilschritte sind von außen beobachtbar. Es wird darauf hingewiesen, dass die Beobachtbarkeit dieses Prozesses dazu beiträgt, den Prozess anschließend einfacher beurteilen zu können.
Innerhalb von kollaborativen Lernszenarien ist die Gruppenzusammensetzung ein weiterer Faktor. Van Dijk et al. (2014) geben einen kurzen Überblick über den diesbezüglichen Stand der Forschung, bevor eine Unterstützungsmöglichkeit des Prozesses untersucht und vorgeschlagen wird. Es wird zunächst die Heterogenität von Gruppen betrachtet. Watson und Marshall (1995) konnten keine signifikanten Leistungsunterschiede in Examen finden, wenn die Teilnehmenden zuvor in leistungshomogenen oder -heterogenen Gruppen arbeiteten. Die Wahrnehmung der ProbandInnen wurde durch Fragebögen erhoben und zeigt ihre Präferenz zu einer homogenen Gruppenzusammensetzung. In einer Studie von Van Dijk et al. (2014) wurden GrundschülerInnen untersucht, die in kollaborativen Szenarien zeichneten und damit Repräsentationen ihres Wissens erzeugten. Sie fanden heraus, dass die Gruppierung nach sozialen Präferenzen einen positiven Einfluss auf die Ergeb-

nisse der Wissenstests der SuS und die Qualität der Zeichnungen hat. Es liegen weitere Studien vor, die die Gruppenzusammensetzung anhand von sozialen Präferenzen betrachten. Azmitia und Montgomery untersuchten den Einfluss von Freundschaft in kollaborativen Lernumgebungen auf das wissenschaftliche Denken. Es wurden FünftklässlerInnen in zwei verschiedene Szenarien aufgeteilt, wobei die einen mit Freunden und die anderen mit Bekannten zusammen arbeiteten. Die befreundeten Paare zeigten dabei eine höhere Genauigkeit beim Problemlösen als die Paare von Bekannten. Es wird Evidenz erbracht, dass befreundete Paare eher dazu tendieren ihre Lösungen zu diskutieren und sich gegenseitig zu kritisieren als bekannte Paare. Ebenfalls wurde gezeigt, dass die Evaluation von Ergebnissen und die Beteiligung an Konflikten mit besserem Problemlösen verbunden sind (Azmitia und Montgomery, 1993). In einer Studie mit Studierenden wurde jedoch auch eine starke negative Korrelation zwischen der Gruppenarbeit unter Freunden und deren erbrachter Leistung festgestellt (Maldonado et al., 2009).
Insgesamt ziehen van Dijk et al. den Schluss, dass die Ergebnisse keinen Konsens aufweisen, ob eine Gruppierung anhand positiver sozialer Präferenz vorgenommen werden sollte. Es muss jedoch darauf hingewiesen werden, dass teilweise relativ unterschiedliche Gruppen und Szenarien betrachtet wurden. Es unterschieden sich die Gruppengrößen, das Alter, die Interventionsdauer, die Zusammensetzung der Gruppen hinsichtlich des Geschlechts und die Methodik der Datengenerierung und -auswertung, um nur einige Variablen zu nennen.

In diesem Abschnitt wurde argumentiert, dass wenige Ergebnisse auf eine negative Auswirkung von Kollaboration auf die Erkenntnisgewinnung von SuS hindeuten. Walpuski und Sumfleth (2007) beschreiben ebenfalls positive Auswirkungen kollaborativer Lernformen, wobei sie diese in Kombination mit Inquiry-Aufgaben als besonders erfolgversprechend einstufen. Die Verwendung kollaborativer Settings erscheint auf Grundlage dieser Ergebnisse für die Untersuchung des Forschungsfelds Physical Computing geeignet zu sein.

2.3 Zusammenfassung

Die existierende Literatur zum Physical Computing zeigt auf, dass inhaltlich und technisch ein großes Potential im Einsatz dieser Geräte als Lernmedium besteht. Es wurden bereits verschiedene Untersuchungen zur Verwendung im Informatikunterricht und MINT-Kontext vorgenommen, welche die Förderung der Motivation der SuS, die Eignung als gendergerechtes Lernmedium und die curriculare Verankerung in Inhalts- und Kompetenzbereichen untersuchten. Die Ergebnisse dazu sind vielversprechend und weisen darauf hin, dass Physical-Computing-Geräte einen Lebensweltbezug für SuS schaffen und Konzepte der Informatik vermittelt werden können. Der Physical-Computing-Begriff schafft keine scharfe Abgrenzung in Gerätekategorien und die Anzahl diverser Geräte wächst rapide. Des Weiteren wurden die meisten Untersuchungen bislang durch selbst-

konstruierte Physical-Computing-Lernszenarien und Testdesigns durchgeführt. Das führt dazu, dass bisherige Studien begrenzt vergleichbar sind und nicht systematisch auf ihnen aufgebaut werden kann. Insbesondere der Wandel innerhalb der Gerätevielfalt und Veränderungen der Anwendungskontexte fanden bislang wenig Eingang in die Forschung. Es lässt sich als konkreter Mangel ableiten, dass bislang zu nah an einzelnen Geräten geforscht wird, die gegebenenfalls in wenigen Jahren durch neue Geräte abgelöst werden. Der Problemlöseprozess, der während der Arbeit mit Physical-Computing-Geräten durchlaufen wird, scheint jedoch weitläufig geräteunabhängig zu sein. Der Physical-Computing-Prozess wird partiell in der Literatur beschrieben und scheint ähnliche Phasen wie der Prozess der wissenschaftlichen Erkenntnisgewinnung der Naturwissenschaften zu umfassen. Letzterer ist ein Prozess, der tiefgründig erforscht wird und dessen Notwendigkeit in nationalen und internationalen Bildungsstandards für die Naturwissenschaften verankert ist. In den Naturwissenschafts-Didaktiken werden zudem die gleichen Geräte (z. B. Mikrocontroller) wie in der Informatikdidaktik verwendet, um Inhalte zu vermitteln. Die Ausstattung durch Sensoren und Aktuatoren scheint per se einen fächerübergreifenden Charakter zu haben, da z. B. die physikalische Funktionsweise von Sensoren für die Arbeit mit ihnen essentiell ist. Gelingt es die Ähnlichkeit beider Prozesse aufzuzeigen, ist naheliegend, dass Erkenntnisse aus der naturwissenschaftlichen Forschung in diesem Bereich auf die Informatik übertragen werden können. Daraus ergibt sich die Frage, welche Gemeinsamkeiten beide Problemlöseprozesse teilen und worin sie sich unterscheiden. Diese Forschungslücke wird wie folgt als Forschungsfrage formuliert:

1. *Welche Gemeinsamkeiten und Unterschiede weisen die Prozesse Physical Computing und die wissenschaftliche Erkenntnisgewinnung auf? Kann Physical Computing als Arbeitsweise der Erkenntnisgewinnung in der Informatik genutzt werden?*

Insbesondere für die Informatikdidaktik kann es einen großen Mehrwert haben, an existierende Forschungsergebnisse anknüpfen zu können. Die Didaktik der Informatik existiert seit circa 30 Jahren und stellt erst seit Kurzem die Frage, was *Nature-of-Computer-Science* beinhaltet (vgl. Schneider und Mühling, 2017). In den letzten Jahren zeichneten sich Schwierigkeiten in der Definition von Informatik als Wissenschaft ab, da sie ingenieurwissenschaftliche und naturwissenschaftliche Züge aufweist (vgl. Denning, 2005). Kompetenzstandards und Testinstrumente sind noch weitgehend in der Entstehungsphase. Im Zuge der Digitalisierung und dem vermehrten Gebrauch von Medien in der Schule, ist jedoch ebenfalls ein Nutzen für eine gemeinsame MINT-Didaktik eine nicht zu vernachlässigende Motivation, für die ein theoretisches Fundament mit dieser Arbeit geschaffen werden kann. Im Rahmen der Literaturrecherche zur 1. Forschungsfrage wurde deutlich, dass der wissenschaftliche Erkenntnisgewinnungsprozess den SuS diverse Schwierigkeiten bereitet. Es handelt sich dabei um ein breites und aktuelles Forschungsfeld, wobei auch die Förderung entsprechender Kompetenzen von SuS in nationalen und internationalen Bildungsstandards gefordert wird. Untersuchungen der wissenschaftlichen Erkenntnisgewinnung zeig-

ten, dass SuS in allen Phasen des Problemlöseprozesses Probleme haben, wie beispielsweise beim Umgang mit Messunsicherheiten oder der Konstruktion eines Versuchsaufbaus. In der Physical-Computing-Literatur gibt es ebenfalls Hinweise darauf, dass die Konstruktion der Geräte neben der Programmierung eine Problemursache sein kann. Auch Messunsicherheiten spielen im Physical Computing eine Rolle, da Umgebungsdaten gemessen, interpretiert und weiterverarbeitet werden. Durch eine konkrete Beschreibung von Problemen und den Problemursachen im Physical-Computing-Prozess wäre es möglich Lehrkräften und SuS Hilfestellungen zu geben, um den Prozess erfolgreich zu durchlaufen. Auch diese Forschungslücke wurde bislang nicht in der Physical-Computing-Forschung adressiert. Deshalb wird die 2. Forschungsfrage dieser Dissertation wie folgt formuliert:

2. *Welche Probleme treten bei der Interaktion mit Physical-Computing-Geräten auf und wie können diese kategorisiert werden?*

Aufgrund der aufgezeigten Probleme von SuS wurden verschiedene analoge und computergestützte Maßnahmen entwickelt, um den Prozess der wissenschaftlichen Erkenntnisgewinnung zu unterstützen. Es wurde gezeigt, dass SuS mit diversen Hilfestellungen bei der Prozessbewältigung bessere Hypothesen aufstellten und Daten gezielt als Grundlage zur Erklärung von Kausalzusammenhängen nutzten. Des Weiteren verbesserte sich die Qualität der aufgestellten Modelle der SuS, wenn sie Unterstützungen zur Selbstregulation in kollaborativen Szenarien erhielten. In der Physical-Computing-Forschung wurden bislang Probleme während des Problemlöseprozesses und mögliche Hilfestellungen kaum betrachtet. Da Physical Computing jedoch als komplexes Problemlösen beschrieben wird sowie Probleme von SuS nur teilweise in der Literatur berücksichtigt werden, ist es die Aufgabe der Informatikdidaktik Hilfestellungen zu entwickeln und diese zu evaluieren. Eine Untersuchung dieser Forschungslücke ergibt sich ebenfalls als logische Konsequenz aus der 1. und 2. Forschungsfrage. Die folgende Forschungsfrage wird in diesem Zusammenhang untersucht:

3. *Welche Arten von Hilfestellungen sind effektiv für Schülerinnen und Schüler, um den Physical-Computing-Prozess zu unterstützen?*

Die identifizierten Forschungslücken bauen damit aufeinander auf und werden im Folgenden durch weitere Literaturrecherchen untersucht.

Kapitel 3

Empirisches Modell des Physical-Computing-Prozesses

Dieses Kapitel beschäftigt sich mit der 1. Forschungsfrage, die im Rahmen dieser Arbeit untersucht werden soll, und baut direkt auf Kapitel 2 auf. Darin wurde die Forschungsfrage aus der Literatur (Analysis and Exploration I des DBR-Prozesses) abgeleitet und wie folgt formuliert:

Welche Gemeinsamkeiten und Unterschiede weisen die Prozesse Physical Computing und die wissenschaftliche Erkenntnisgewinnung auf? Kann Physical Computing als Arbeitsweise der Erkenntnisgewinnung in der Informatik genutzt werden?

Zur Beantwortung dieser Forschungsfrage wird zunächst ein Modell des Physical-Computing-Prozesses auf Literaturbasis erstellt, um es anschließend mit einem Modell der naturwissenschaftlichen Erkenntnisgewinnung vergleichen zu können (Kapitel 3.1). Dadurch wird die theoretische Grundlage für folgende Untersuchungen geprüft. Kapitel 3.2 bildet die Phase Design and Construction I und beinhaltet Methoden und Instrumente, die der Klärung der 1. Forschungsfrage dienen. Das Vorgehen der Untersuchung wird darin geplant.
Im Anschluss wird eine empirische Untersuchung vorgestellt (entspricht der Phase Evaluation and Reflection I), die das Ziel verfolgt, das zuvor vorgestellte theoretische Modell in seinen Bestandteilen zu überprüfen und den Verlauf des Physical-Computing-Prozesses herauszuarbeiten (Kapitel 3.3). Somit soll gezeigt werden, ob das theoriegeleitete Physical-Computing-Modell durch Empirie gestützt werden kann und Physical Computing als eine Arbeitsweise der Erkenntnisgewinnung genutzt werden kann.
Es wird die herausgestellte Forschungslücke adressiert, dass bisherige Studien im Physical Computing aufgrund verschiedener Gerätetypen, Aufgabenstellungen und nicht validierter Testinstrumente nicht vergleichbar sind. Eine tiefgründige Untersuchung eines allgemeingültigen Physical-Computing-Prozesses schafft einerseits eine Ebene der Vergleichbarkeit, da als Grundlage ein gemeinsames Modell genutzt werden kann. Dadurch können

Forschungsergebnisse in das Modell eingeordnet werden, z. B. besondere Herausforderungen zu konkreten Phasen des Prozesses. Des Weiteren können aus dem Modell heraus Studien konzipiert werden. Andererseits kann die Physical-Computing-Forschung nachhaltiger umgesetzt werden, wenn sie von einzelnen Geräten losgelöst wird. Bestandteile des Modells werden bereits von verschiedenen Autoren im Bereich des Physical Computing beschrieben. Die theoretische Grundlage für die Verknüpfung mit naturwissenschaftlichen Modellen wurde in den Kapiteln 2.1 und 2.2 gelegt. Zusätzlich besteht die Bedeutung eines Prozessmodells des Physical Computing für die Informatikdidaktik darin, dass eine Strukturierung des Prozesses eine genauere Beschreibung von Problemen, dem Zeitpunkt des Auftretens von Problemen und Ansatzpunkten für eine Intervention ermöglicht.

3.1 Theoriegeleiteter Vergleich der Prozesse des Physical Computing und der wissenschaftlichen Erkenntnisgewinnung

In diesem Abschnitt sollen existierende Forschungsbeiträge aufgezeigt werden, die bereits die wissenschaftliche Erkenntnisgewinnungsforschung mit der Robotik bzw. dem Physical Computing in Verbindung bringen. Zuvor wird ein Modell vorgestellt, welches aus der Physical-Computing-Literatur abgeleitet wird und als Grundlage für Vergleiche unter den Modellen dient.
Es wurde ein Prozessmodell (siehe Abbildung 3.1) konstruiert, das die Phasen beim Lösen von Physical-Computing-Aufgaben beschreibt (Schulz und Pinkwart, 2016). Es basiert auf der zuvor vorgestellten Literaturrecherche im Bereich des Physical Computing (vgl. Kapitel 2.1, O'Sullivan und Igoe; Okita; Banzi), wobei die Beiträge daraufhin untersucht wurden, ob sie konkrete Aktivitäten von SuS beschreiben, die Rückschlüsse auf den ablaufenden Prozess des Physical Computing zulassen. Anschließend wurde eine weitere Literaturrecherche bezüglich expliziter Prozessbeschreibungen im Physical Computing vorgenommen, um den Detailgrad des Modells zu erhöhen. Die von den oben genannten AutorInnen beschriebenen Teilphasen wurden aus der Literatur extrahiert und in einem Prozessmodell angeordnet. Jedoch gehen die einzelnen Beiträge zumeist nicht auf alle Phasen des Prozesses ein, sondern beschreiben zum Teil nur eine Haupt- oder Teilphase aus dem hier vorgestellten Modell.

Erste Hinweise auf eine mögliche Verknüpfung des Physical-Computing-Prozesses mit dem Prozess der wissenschaftlichen Erkenntnisgewinnung befinden sich in Aussagen wie: „The essential characteristics of the maker sensibility - deep engagement with content, experimentation, exploration, problem-solving, collaboration, and learning to learn - are the very ingredients that make for inspired and passionate STEM learners“ (Honey und Kanter, 2013, S. 4). An dieser Stelle wird das Maker Movement konkret aufgegriffen und in

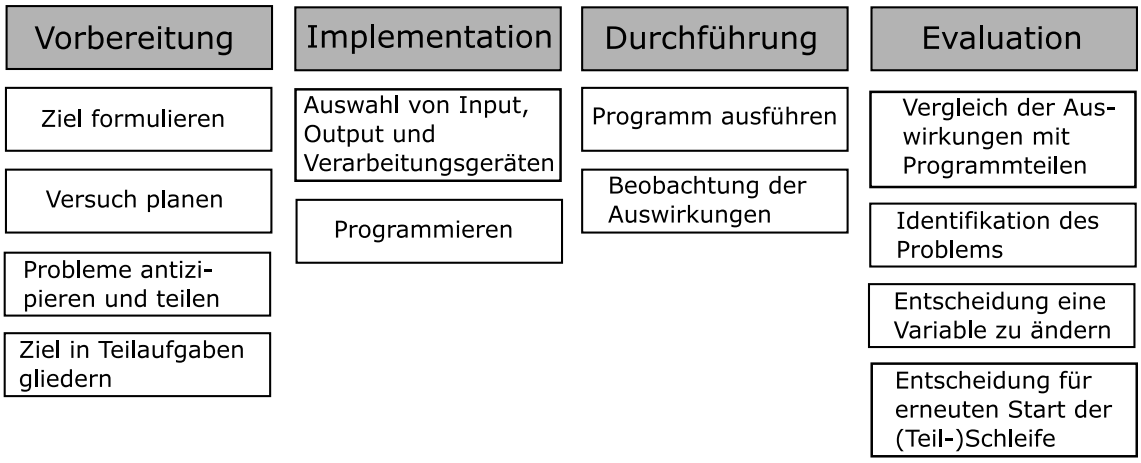


Abbildung 3.1: Physical-Computing-Modell nach Schulz und Pinkwart (2016)

Kontext mit Lernprozessen gesetzt, z. B. dem Experimentieren und dem Problemlösen in STEM (ein Akronym für *Science*, *Technology*, *Engineering* und *Mathematics*).
Ein Ansatz zur Verknüpfung von Robotik und Scientific Inquiry wurde von Sullivan auf der Ebene von beobachteten Fertigkeiten vorgenommen (Sullivan, 2008). Ziel des Papers ist, die Beziehung von Robotik-Aktivitäten und Science-Literacy-Fertigkeiten herzustellen. Deshalb wurde untersucht, wie SuS im Alter von 11 - 12 Jahren Denkweisen und prozessbezogene Fertigkeiten nutzten, um Robotik-Aufgaben zu lösen. Zusätzlich wurde abschließend der Lernzuwachs der 26 Teilnehmenden ermittelt. Die Untersuchung fand im Rahmen eines dreiwöchigen Intensivkurses während des Sommers statt. Die Autorin argumentiert, dass die Ziele von Science Literacy, die von NRC (National Research Council, 1996, S. 23) und AAAS (American Association for the Advancement of Science and others, 1993) definiert wurden, eine starke Verbindung zu denen der Robotik aufweisen. Dabei führt Sullivan folgende Überschneidungspunkte auf:

1. Eine große Schnittmenge von Denkweisen (Berechnen, Vermutungen anstellen, Manipulation, Beobachtung) ist vorhanden.

2. Scientific Inquiry wird durch technisches Design und Computerprogrammierung gefördert.

3. Robotik ist ebenfalls ein Lerngegenstand für Systeme.

Videographie und eine anschließende Transkription bilden die Grundlage für Sullivans Untersuchung. Durch die Beobachtungen wurden die definierten Denkweisen (Beobachtung, Vermutungen anstellen, Manipulation und Berechnen) und prozessbezogenen Fertigkeiten (Evaluation von Lösungen, Hypothesengenerierung, Hypothesen testen und die Kontrolle von Variablen) ermittelt. Die Phase der Beobachtung nimmt mit 31 % den größten Anteil ein. Auffällig ist, dass die Evaluation von Lösungen den zweitgrößten Anteil aller codierten Phasen einnimmt. Die Evaluation wurde mit einer Häufigkeit von 19 % aller codierten

Phasen vorgenommen. Sie definierte diese Phase mit „[the] student makes an evaluative comment about the solution“ (Sullivan, 2008, S. 379). Durch den verwendeten Pre- und Posttest resultiert ebenso, dass die SuS ihr Systemverständnis durch die Teilnahme am Robotikkurs verbessert haben. Anhand der Testitems ist anzunehmen, dass sich das Systemverständnis ausschließlich auf Robotik und nicht auf Naturwissenschaften bezieht. Die Items umfassen ausschließlich Aufgaben im Bereich von Informatiksystemen und der Datenverarbeitung.

Über die Verbindung des Physical-Computing-Prozesses hinaus, wurde der wissenschaftliche Erkenntnisgewinnungsprozess der Naturwissenschaften mit dem Problemlösen in den Disziplinen Technologie und Engineering verglichen. In den zuletzt genannten Disziplinen wird das Problemlösen als sogenannter *Engineering-Prozess* bezeichnet.
In der Literatur wird dargestellt, dass der Designprozess von technologischen Domänen parallel zur wissenschaftlichen Erkenntnisgewinnung verläuft (National Research Council, 1996). Eine detaillierte Analyse wird von Lewis (2006) vorgenommen, bei der Gemeinsamkeiten und Unterschiede beider Prozesse herausgestellt werden. Dabei kommt er zu dem Schluss, dass die Unterschiede (siehe S. 52 unten) zwischen Naturwissenschaften und Technik dazu verwendet werden können, beide Prozesse zusammenhängend in die Schule zu integrieren. Es ist demzufolge möglich, das Verständnis über Naturwissenschaften durch Design-Aktivitäten zu verbessern, was ebenfalls zur Verbesserung des Designs führt. Für diese Integration müssen jedoch verschiedene institutionelle Rahmenbedingungen berücksichtigt werden.
Als Gemeinsamkeit wird herausgestellt, dass es sich um wissenschaftliches Denken handelt und beide Prozesse das Ziel verfolgen, von einem Problem zu einer Lösung zu kommen. Auch beschäftigen sich beide Prozesse mit dem Testen, Evaluieren und dem darauf basierten Treffen von Entscheidungen. Sie unterliegen dabei diversen Bedingungen, z. B. Fragen der Sicherheit und ethischen Überlegungen.
Die dargestellten Unterschiede sind bereits in den Bedingungen zu finden, die an den jeweiligen Prozess gestellt werden. Lewis zieht den Schluss, dass die Bedingungen für den Engineering-Prozess, z. B. Performance, Größe, Wert und Sicherheit (vgl. Kroll et al., 2001), nicht mit denen der Naturwissenschaften vergleichbar sind. Auch im Bereich des Designs von Produkten werden Abwägungen getätigt, die in den Naturwissenschaften vernachlässigt werden können, z. B. zwischen Funktionalität und Ästhetik. Auch in Bezug auf die praktische Durchführung von naturwissenschaftlichen Experimenten müssen kaum kommerzielle Abwägung betrachtet werden. Als größter Unterschied zwischen beiden Prozessen wird der Zweck der wissenschaftlichen Erkenntnisgewinnung identifiziert. In den Naturwissenschaften wird dieser als explorativ beschrieben, um ein Phänomen zu entdecken oder zu verstehen. Im Design hingehen sollen künstliche Erzeugnisse geschaffen werden, die diese dann durch ihren Eingriff in die Umgebung verbessern.
Aufgrund der beschriebenen Bedingungen für den Engineering-Prozess, scheint dieser un-

geeignet für einen Vergleich mit dem Physical-Computing-Prozess zu sein. Jedoch können Abwägungen zwischen Funktionalität und Ästhetik sowie von anfallenden Kosten eine Rolle bei Physical-Computing-Aktivitäten spielen. Der Ansatz von Sullivan (2008) hingegen, der auf die Ähnlichkeit von Fertigkeiten von SuS für beide Prozesse fokussiert, erscheint für schulische Kontexte in Verbindung zum Physical Computing besonders relevant zu sein.
Bei dem Vergleich muss beachtet werden, dass STEM und MINT sich in ihrer zugewiesenen Rolle der Informatik unterscheiden, obwohl sie zumeist synonym betrachtet werden. In dem Akronym STEM wird die Informatik durch einen Querschnitt aus *Technology* und *Engineering* repräsentiert. Der deutsche Term MINT beschreibt hingegen *Mathematik*, *Informatik*, *Naturwissenschaften* und *Technik*, so dass die Informatik ein expliziter Bestandteil ist. De Vries verweist darauf, dass das „I“ ebenfalls den Bereich *Ingenieurswissenschaften* adressieren kann und damit ein Äquivalent zu STEM bilden würde (De Vries, 2016). Aus dieser Darstellung wird ersichtlich, dass die Rolle der Informatik in MINT bzw. STEM hinterfragt werden sollte, bevor die Begriffe inflationär benutzt werden.

Auf Grundlage der aufgezeigten Ergebnisse innerhalb der Literaturrecherche (Kapitel 2.1, 2.2) werden nun die Phasen des Experimentierprozesses denen des Physical Computing gegenübergestellt, um Gemeinsamkeiten und Unterschiede dieser zu verdeutlichen. Der Experimentierprozess ist ein Vertreter der Arbeitsweisen zur Erlangung von wissenschaftlichen Erkenntnissen. Für diesen Vergleich wird das zuvor konstruierte Modell des Physical-Computing-Prozesses (vgl. Abbildung 3.1) genutzt. In Tabelle 3.1 sind nur die Phasen abgebildet, die explizit in den Modellen aus der Literatur modelliert sind, wobei Abweichungen des idealtypischen Prozesses möglich sind. Diese Gegenüberstellung wurde bereits in Schulz und Pinkwart (2016) publiziert, wobei im Rahmen dieser Arbeit marginale Veränderungen vorgenommen wurden. In der Tabelle werden in der 1. und 4. Spalte die Bezeichnung der Hauptphasen für den Experimentierprozess bzw. den Physical-Computing-Prozess angegeben. In der 2. und 3. Spalte werden den Hauptphasen Teilphasen zugeordnet, in denen sie auftreten können. Sofern möglich, werden sie auf der gleichen Ebene (bzw. in der gleichen Zeile) wie ihr Pendant aus dem jeweils anderen Prozess angeordnet. Befindet sich auf einer Ebene nur eine Phasenbeschreibung, so kann dieser Teilphase kein Gegenüber des anderen Prozesses zugeordnet werden.
Beide dargestellten Prozesse sind in die Phasen *Vorbereitung*, *Durchführung* und *Evaluation* gegliedert. Beim Prozess des Physical Computing ist nach der *Vorbereitungsphase* zusätzlich eine Phase der *Implementation* enthalten. Im Experimentierprozess sind außerdem zwei Teilphasen enthalten, die sich keiner der drei Hauptphasen zuordnen lassen, da es sich um eine Schnittstelle von zwei Hauptphasen handelt.

Tabelle 3.1: Vergleich des Experimentierprozesses der wissenschaftlichen Erkenntnisgewinnung nach Schreiber et al. (2009) mit dem Physical-Computing-Prozess gemäß O'Sullivan und Igoe (2004); Banzi (2011); Okita (2014)

Phase	**Experimentier-Prozess**	**PhC-Prozess**	**Phase**
Vorbereitung	Vorgegebene Fragestellung klären/ Fragestellung entwickeln	Ziel formulieren	**Vorbereitung**
	Erwartungen formulieren	Versuch planen	
	Hypothese bilden		
		Ziel in Teilaufgaben gliedern	
		Probleme antizipieren und teilen	
Vorb./ Durchf.	Versuchsplan entwerfen	Auswahl von Input, Output und Verarbeitungsgeräten	**Implementation**
Durchführung	Geräte zusammenstellen		
	Versuchsanordnung aufbauen		
		Programmieren	
	Messungen durchführen	Programm ausführen	**Durchführung**
	Messungen dokumentieren	Beobachtung der Auswirkungen	
Durchf./ Evaluation	Umgang mit Problemen und Fehlern		
Evaluation	Messdaten aufbereiten		**Evaluation**
	Messdaten verarbeiten		
	Ergebnis interpretieren		
		Vergleich der Auswirkungen mit Programmteilen	
		Identifikation des Problems	
		Entscheidung eine Variable zu ändern	
		Entscheidung für erneuten Start der (Teil-)Schleife	

In beiden Prozessen bildet die *Vorbereitungsphase* den Ausgangspunkt des Problemlösens. Diese Phase wird zur Klärung oder Entwicklung einer Frage bezüglich eines beobachteten Phänomens genutzt. Im Physical Computing wird ein relativ klares Ziel gesetzt, das verfolgt werden soll bzw. handelt es sich um ein konkretes Problem, das gelöst werden soll. Anschließend wird in beiden Prozessen eine zu testende Hypothese formuliert bzw. ein Lösungsweg konstruiert. Darauf aufbauend müssen Erwartungen bezüglich der Ergebnisse formuliert werden. Nur im Physical-Computing-Prozess wird explizit modelliert, dass schon in der Vorbereitung die Aufgabe in Teilaufgaben aufgegliedert werden sollte. Außerdem wird empfohlen, Probleme zu antizipieren und gegebenenfalls in der entsprechenden Community zu teilen, z. B. in Internetforen.

Der Übergang von der Vorbereitung zur praktischen Durchführung gestaltet sich beim Experimentieren dadurch, dass zunächst ein experimentelles Design erstellt wird. Klar zugeordnet zur *Durchführungsphase* ist die konkrete Auswahl von Geräten und verwendeten Materialien, sowie der Versuchsaufbau. Alle diese Teilphasen des Experimentierens entsprechen im Physical-Computing-Prozess der Auswahl von Input, Output und Verarbeitungsgeräten, was den Aufbau inkludiert. Das Programmieren im Physical Computing ist Bestandteil der *Implementationsphase* und hat kein direktes Pendant im Experimentierprozess. Nach der Implementationsphase sind die beiden Prozesse wieder sehr ähnlich, indem Messungen aufgenommen und dokumentiert werden. Beim Physical-Computing-Prozess werden das geschriebene Programm ausgeführt und die Auswirkungen beobachtet. Hier unterscheiden sich die Prozesse darin, wie mit dem Ergebnis umgegangen wird. Beim Experimentieren werden die Daten dokumentiert, da im Anschluss ein strukturierter Auswertungsprozess erfolgt. Dies ist im Physical-Computing-Prozess keine gängige Vorgehensweise.

Innerhalb des Experimentierprozesses wird explizit darauf hingewiesen, dass mit Problemen und Fehlern umgegangen werden muss. Dies ist in einem Übergang von der Durchführungs- zur Evaluationsphase verortet. In der *Evaluationsphase* unterscheiden sich beide Prozesse am meisten. Beim naturwissenschaftlichen Experimentieren werden die Rohdaten zunächst vorbereitet und verarbeitet, bevor sie auf die Hypothese hin interpretiert werden können. Beim Physical-Computing-Prozess hingegen werden die Abweichungen des geschriebenen Programms und der Auswirkungen verglichen, um anschließend das Problem (zur Erreichung des Zielzustands) zu identifizieren. Da eine Lösung des Problems beim ersten Durchlauf unwahrscheinlich erscheint, wird explizit darauf hingewiesen, dass dieser Prozess als Schleife erneut gestartet werden sollte. Aus diesem Grund können weitere Durchläufe mit geänderten Variablen hilfreich sein.

Der Ausgangspunkt des Physical-Computing-Prozesses ist die Formulierung eines konkreten Ziels. Da diese Teilphase an den Engineering-Prozess erinnert, scheint an dieser Stelle der Physical-Computing-Prozess dem Engineering-Prozess ähnlicher zu sein als dem der wissenschaftlichen Erkenntnisgewinnung. Diese Ähnlichkeit lässt sich darauf zurückführen, dass es sich beim Physical Computing nicht zwangsläufig um ein beobachtetes Problem

oder Phänomen aus der Umgebung handelt. Vielmehr stellt man sich diese Aufgaben selbst, um sich z. B. den Alltag zu erleichtern. Dies wird auch in dem folgenden Zitat verdeutlicht: „As used in the Standards, the central distinguishing characteristic between science and technology is a difference in goal: The goal of science is to understand the natural world, and the goal of technology is to make modifications in the world to meet human needs. Technology as design is included in the Standards as parallel to science as inquiry. Technology and science are closely related. A single problem often has both scientific and technological aspects“ (National Research Council, 1996, S. 24).
Sofern der erste Durchlauf des Prozesses stattgefunden hat und evaluiert wurde, ist jedoch die Nähe zur naturwissenschaftlichen Erkenntnisgewinnung ersichtlich. Die SuS stoßen dabei zumeist auf Probleme, die sie anschließend in Teilprobleme zerlegen. Die Teilprobleme werden in weiteren Iterationen des Inquiry-Prozesses bearbeitet. Zumeist werden dafür Erkenntnisse über das System oder die Untersuchung von beobachteten Phänomenen benötigt, um das ursprüngliche Ziel zu erreichen. Das könnte z. B. die Ermittlung von Messgrenzen eines Sensors oder das Prinzip der Eingabe-Verarbeitung-Ausgabe (EVA) in der Informatik sein.

In der Literatur sind vielversprechende Beschreibungen von der Überschneidung des Physical-Computing-Prozesses und dem Prozess der wissenschaftlichen Erkenntnisgewinnung zu finden. Jedoch werden sie zumeist auf einer allgemeinen Ebene und nicht auf der Ebene von Teilphasen betrachtet. Der vorgenommene Vergleich von Teilphasen lässt ebenfalls darauf schließen, dass beide Prozesse einige Gemeinsamkeiten teilen. Auf dieser Grundlage wird im Folgenden eine empirische Untersuchung des Physical-Computing-Prozesses vorgenommen. Somit kann anschließend ein zuverlässigerer Vergleich in Hinblick auf die praxisrelevante Aspekte des Physical-Computing-Prozesses ermöglicht werden.

3.2 Methoden und Instrumente I

Für die Untersuchung der 1. Forschungsfrage werden verschiedene Methoden und Testinstrumente benötigt, die in diesem Abschnitt vorgestellt werden. Sie bilden den Ausgangspunkt für die durchgeführten Studien. Im DBR-Prozess ist dieser Abschnitt der Phase Design and Construction I zuzuordnen, da das Untersuchungsdesign der 1. Forschungsfrage, zugehörige Aufgabenstellungen und ein Codiermanual entwickelt werden.

3.2.1 Methoden I

In diesem Abschnitt werden *Leitfadeninterviews*, *Videostudien* und *qualitative Inhaltsanalysen* als verwendete Methoden kurz vorgestellt.
Die hier getroffene Auswahl setzt sich aus etablierten Methoden der Naturwissenschafts-Didaktik und Kognitionspsychologie zusammen. Eine detaillierte Betrachtung der methodischen Umsetzung erfolgt in den Kapiteln, in denen die Methoden zur Anwendung kommen.
Ausgehend von der Forschung des naturwissenschaftlichen Problemlösens und damit verbundenen Fragestellungen, sind ausgewählte Methoden gängig und werden in der Literatur immer wieder verwendet. Nach Mayer (2007) lassen sich diese zusammenfassen zu: Laborexperimenten, Klassenraumstudien, Interviews und Fragebögen.
Abschließend wird die Datenerhebung für die folgenden Studien beschrieben.

Leitfadeninterview

In der naturwissenschafts-didaktischen Forschung ist das Leitfadeninterview ein wichtiges qualitatives Instrument. Es dient der Datenerhebung, ohne jedoch den Gesprächsfluss zu stören. Die befragende Person lenkt dabei das Gespräch auf Themenbereiche oder erlebte Situationen, um an die Gefühle oder Vorstellungen von der interviewten Person zu gelangen (Niebert und Gropengießer, 2014). In Anlehnung an Flick (2007) werden dabei verschiedene Interviewtypen unterschieden. Im Rahmen dieser Arbeit wird vor allem das *fokussierte Interview* nach Merton (1946) eine Rolle spielen. Dieser Interviewtyp wird eingesetzt, um bereits erlebte Situationen zu reflektieren, z. B. ein durchgeführtes Experiment. Zumeist wird dieser Interviewtyp als Gruppengespräch durchgeführt, Einzelgespräche werden jedoch nicht ausgeschlossen. Der jeweilige Interviewtyp wird abhängig davon ausgewählt, welche Daten durch das Instrument erhoben werden sollen. Bei dem fokussierten Interview stehen die Emotionen, Interessen und Vorstellungen der Befragten im Vordergrund. Ein Vorteil besteht darin, dass das Leitfadeninterview eine Struktur vorgibt, jedoch auch jederzeit davon abgewichen und auf Antworten reagiert werden kann, z. B. durch gezieltes Nachfragen.

Videostudien

Bei einer Videostudie handelt es sich um eine oft genutzte Forschungsmethode zur Analyse von Unterricht und anderen Lernszenarien. Sie stellt eine breite Datenbasis zur Verfügung, die nach der Aufzeichnung vielseitig weiterverarbeitet werden kann (Brückmann und Duit, 2014). Zumeist werden die Videos verschriftlicht, z. B. mit Hilfe einer Transkriptionssoftware. Dabei wird das Gesprochene niedergeschrieben und je nach Ziel der Studie, Gestiken und Handlungen der gefilmten Personen hinzugefügt. Dieser Schritt hat außerdem das Ziel, die Daten zu anonymisieren, indem personenbezogene Daten entfernt werden. Um das Datenmaterial zu analysieren und zu kategorisieren, wird anschließend eine Analysesoftware verwendet, z. B. MaxQDA (VERBI Software. Consult. Sozialforschung. GmbH, 2017). Bei der Auswertung des Videomaterials wird zwischen zeit- und eventbasierter Auswertung unterschieden. Letzteres bietet sich besonders an, wenn Aussagen von SuS codiert werden sollen.
In den durchgeführten Studien wurden die Videos mittels MaxQDA 12 codiert. Dabei wurden die Gespräche der SuS sowie das Gesprochene der Lehrkraft, Aktionen der SuS (Gestiken und Veränderungen in der Programmieroberfläche sofern sichtbar) und die Aktionen von programmierten Geräten transkribiert. Die zugrundeliegenden Transkriptionsregeln sind in Anhang E dargestellt. Verwendete Transkriptionsmanuale folgen u. a. in Abschnitt 3.2.2.

Qualitative Inhaltsanalyse

In den Leitfadeninterviews und Videostudien werden qualitative Daten aufgenommen, die anschließend durch die qualitative Inhaltsanalyse weiterverarbeitet und somit ausgewertet werden können. Ein Vorteil dieses Ansatzes ist es, dass die Daten bei der Weiterverarbeitung dauerhaft in qualitativer Form vorliegen, jedoch trotzdem quantifiziert werden können (Mayring, 2010). Der Informationsverlust soll somit gering gehalten werden. Die qualitative Inhaltsanalyse adressiert eine systematische und regelbasierte Analyse von Texten, Bildern oder ähnlichem Material, sofern diese Daten protokolliert sind. Zur Analyse wird ein Kategoriensystem oder Codesystem aufgestellt, das anhand von in Literatur beschriebener Kategorien deduktiv gebildet wird. Es wird induktiv auf Grundlage des vorliegenden und zu analysierenden Protokolls ergänzt. Um die Kategorien auf den Einzelfall anzupassen, wird dieses Zusammenspiel iterativ mit den vorliegenden Daten durchlaufen. Nach der Fertigstellung des Kategoriensystems werden die Protokolle auf dieser Grundlage codiert (Mayring, 2000).
Die qualitative Inhaltsanalyse unterliegt einem hohen Maß an Systematik, was gerade im wissenschaftlichen Kontext die Nachvollziehbarkeit der Analyse und der resultierenden Ergebnisse steigert. Des Weiteren liegt ein großer Vorteil darin, dass quantitative Schritte in den Prozess eingebracht werden können (Mayring, 2010).
Zur Sicherstellung der Reliabilität eines Codesystems kann die sogenannte *Interrater-*

Reliabilität ermittelt werden. Dabei recodiert z. B. eine weitere Person die gleiche Datenbasis (bzw. einen Teil davon) mit dem gleichen Kategoriensystem, woraus anschließend die Übereinstimmung der Codierer ermittelt wird (Hammann und Jördens, 2014). Abhängig von der Art der Daten, existieren verschiedene statistische Maße zur Berechnung der Interrater-Reliabilität. In dieser Arbeit wird *Cohens Kappa* verwendet, da es die Übereinstimmung von zwei Codierern und nominalskalierten Daten adressiert. Die Cohens-Kappa-Werte werden in folgenden Wertebereichen beurteilt als: < 0.00 „schlecht“, $0.00 - 0.20$ „leicht“, $0.21 - 0.40$ „ausreichend“, $0.41 - 0.60$ „moderat“, $0.61 - 0.80$ „substantiell“ und $0.81 - 1.00$ „fast perfekt“ (Landis und Koch, 1977).

3.2.2 Instrumenten I

Ein wichtiger Bestandteil von empirischer Forschung sind Instrumente, die in durchgeführten Interventionen genutzt werden. Um die Nachvollziehbarkeit der Studien zu sichern, werden in diesem Kapitel alle Aufgabenstellungen dargestellt, welche die SuS in den Studien bearbeiten und lösen sollen. Da bereits aus der wissenschaftlichen Erkenntnisgewinnung bekannt ist, dass SuS diverse Hilfen benötigen, um einen Problemlöseprozess erfolgreich zu durchlaufen, bekommen die SuS zusätzlich zu den Aufgaben eine Hilfestellung angeboten.
Die Auswertung von qualitativen Daten wird durch eine qualitative Inhaltsanalyse vorgenommen. Dafür wird ein Codiermanual erstellt und für die Beantwortung der Forschungsfragen genutzt. Der Entwicklungsprozess des Codiermanuals wird im Rahmen der Bearbeitung der 1. Forschungsfrage vorgenommen. Das resultierende Codiermanual wird bereits in diesem Kapitel detailliert beschrieben.

Physical-Computing-Aufgaben I

Die SuS bekommen Aufgaben, um diese mit Physical-Computing-Geräten zu lösen. Inhaltlich orientieren sich diese Aufgaben an Standardaufgabenstellungen bei der Verwendung von LEGO-Robotern[1] oder Arduino-Mikrocontrollern[2]. Da diese Aufgabenstellungen sich schon in zahlreichen Kursen mit SuS als guten Einstieg und leicht verständlich erwiesen haben, sollten sie die Studienergebnisse hinsichtlich des Physical-Computing-Prozesses nicht verfälschen. Bei den Studien $1 - 3$ werden die Aufgaben in Tabelle 3.2 verwendet. Alle Gruppen sollen die ersten beiden Aufgaben lösen, während die 3. Aufgabe als Zusatzaufgabe für besonders schnelle Gruppen fungierte.
In der 1. Aufgabe sollen sich die SuS mit den physikalischen Grenzen des Ultraschallsensors auseinandersetzen. Dieses Wissen wird benötigt, um anschließend geeignete Software zu implementieren bzw. das Physical-Computing-System daraufhin anzupassen. Darüber

[1] vgl. https://le-www-live-s.legocdn.com/sc/media/files/ev3-introduction-to-robotics/introduction-to-robotics-tablet-de-c63bea50480fcd69c499ba365352e55c.pdf?la=en-au
[2] vgl. https://www.sunfounder.com/learn/category/basic-kit-for-arduino.html

Tabelle 3.2: Aufgaben bei der Arbeit mit LEGO Mindstorms-Robotern I

1. Erprobt, welche physikalischen Grenzen der Ultraschallsensor besitzt.
2. Programmiert den Roboter so, dass er um die Kiste fahren kann, alleinig unter der Nutzung des Ultraschallsensors.
3. Programmiert den Roboter so, dass er in der Lage ist, ein vorgegebenes Labyrinth zu durchfahren.*

*Lediglich eine Gruppe aus den Studie 1 - 3 bearbeitete und löste die 3. Aufgabe.

hinaus kann durch diese Aufgabe beobachtet werden, wie die SuS ein physikalisches Experiment durchführen. In der 2. Aufgabe ist ein erster Algorithmus zu implementieren. Dafür werden Kontrollstrukturen wie Schalter (*If-Else-Bedingungen*) und Schleifen (bspw. *Do-While-Schleifen*) sowie die Unterscheidung von Input/Output und serieller/paralleler Prozesse in Programmen benötigt. Ein möglicher Algorithmus beschreibt, wie der Roboter kontinuierlich geradeaus fährt und regelmäßig überprüft (sich dreht), ob die Kiste noch neben ihm ist. Wenn ja, dann fährt er kurz rückwärts, dreht sich zurück und fährt wieder einige Zentimeter geradeaus. Wenn nein, fährt er direkt einige Zentimeter gerade aus. Es ist hilfreich, eine feste Richtung vorzugeben, die der Roboter um die Kiste fahren soll und ihn während des gesamten Algorithmus immer wenige Grad mehr zu dieser Richtung drehen zu lassen als in die Gegenrichtung.
Die 3. Aufgabe kann für die meisten Labyrinthe mit dem identischen Algorithmus wie in Aufgabe 2 gelöst werden, in dem der Roboter an einer Wand entlangfährt. Abhängig von der Größe des Labyrinths müssen ggf. kleine Veränderungen vorgenommen werden, z. B. die Größe der Drehungen und die Weite des Fahrens.

In der 4. Studie arbeiten die SuS mit Arduino-Mikrocontrollern und bekommen die folgenden Aufgaben gestellt (Tabelle 3.3).

Tabelle 3.3: Aufgaben bei der Arbeit mit Arduino-Mikrocontrollern

1. Konstruiert und programmiert eine Ampel aus drei LEDs und einem Lichtsensor, die *rot* leuchtet, wenn der Sensor verdeckt ist, *gelb* bei Umgebungshelligkeit und *grün*, wenn der Sensor mit einer Taschenlampe angestrahlt wird.
2. Konstruiert und programmiert mit Hilfe eines Servomotors ein System, das in der Lage ist, eine Lichtquelle zu suchen und bei dieser stehen zu bleiben.**

**Lediglich zwei der fünf Gruppen bearbeiteten die 2. Aufgabe.

Für die Bearbeitung der 1. Aufgabe mit Arduino-Mikrocontrollern benötigen die SuS ebenfalls Kenntnisse über Kontrollstrukturen (Schleifen, Schalter), Input/Output und serielle/parallele Prozesse. Um die Aufgabe zu lösen, dass unterschiedliche LED-Lampen in Abhängigkeit von der Umgebungshelligkeit aufleuchten, müssen die Intervallgrenzen für die einzelnen Fälle sorgfältig ermittelt und wechselnde Umgebungsfaktoren einbezogen werden. Kenntnisse über die Konstruktion von Stromkreisen sind ebenfalls notwendig.
In der 2. Aufgabe stellt die Konstruktion eine Herausforderung dar, da der Sensor an dem Motor befestigt werden kann und der Motor selbst fixiert werden muss, damit die mechanische Energie gezielt auf andere Komponenten übertragen werden kann. Die Ansteuerung des Servomotors kann ebenfalls problematisch sein, da eine Gradzahlen für die Richtung gezielt eingestellt werden muss. Dafür ist zunächst zu ermitteln, wie der Motor kalibriert ist und für welche Positionen die Angaben *clockwise* und *anti-clockwise* gelten.

Beschreibung der Hilfestellung *4-Phasen analog* Für die Studien 1, 2 und 4 wird als Unterstützung des Physical-Computing-Prozesses eine grobe Strukturierung des Prozesses vorgegeben. Die SuS erhalten verschiedene Aufgaben und werden dazu angehalten, während der Aufgabenbearbeitung jeweils ein viergeteiltes Arbeitsblatt auszufüllen (siehe Anhang B.1). Während sie den Prozess durchlaufen, haben sie für jede Aufgabe das gleiche Blatt zur Verfügung, auf dem sie die freien Felder vervollständigen können. Das erste Feld erwartet die Entwicklung einer Hypothese vor Beginn der Aufgabenbearbeitung. Anschließend soll beschrieben werden, wie die Lösung als Programm implementiert werden kann. Das dritte Feld erfragt die Beschreibung von getätigten Beobachtungen während der Ausführung des Programms auf dem Roboter. Abschließend sollen diese im vierten Feld evaluiert werden.

In den Studien konnte beobachtet werden, dass die SuS die Hilfestellung *4-Phasen analog* kaum nutzten und zumeist nur einen Durchlauf des Prozesses beschrieben. Alle TeilnehmerInnen der 1. Studie füllten das Blatt zur Aufgabe *Kiste umfahren* aus. In Anhang C.1 und C.2 sind die original ausgefüllten Blätter von SuS der 1. Studie zu finden. Dabei wird deutlich, dass sie die Arbeitsblätter nicht effektiv als Strukturhilfe nutzten. Die SuS der 2. Studie füllten keinen der Zettel aus und von den fünf Gruppen der 4. Studie füllten zwei das Arbeitsblatt zur 1. Aufgabe aus (in Anhang C.3). Die zweiten Gruppe der 4. Studie füllte ihre Arbeitsblätter nicht mit Inhalten aus, die für die Studie relevant waren.

Codiermanual

Im Rahmen der qualitativen Inhaltsanalyse werden Transkripte auf verschiedene Codes untersucht. Die Definition der Codes wird im Codiermanual festgehalten, auf dessen Grundlage eine Analyse der Transkripte vorgenommen wird.
In Tabelle 3.4 ist die deduktive Beschreibung von Kategorien dargestellt, die in Kapitel 3.3 genutzt werden, um die 1. Forschungsfrage zu untersuchen. Eine detaillierte Auflistung von

induktiv abgeleiteten Kategorien ist in Tabelle 3.5 dargestellt.

Tabelle 3.4: Deduktive Kategorienbeschreibung zur Forschungsfrage 1 (Analyse des Physical-Computing-Prozesses)

Name (Abkürzung)	**Beschreibung**	**Beispiel**
Vorbereitungsphase (V)	Umfasst alle Aussagen zur allgemeinen Funktion der Technologien und zur Erreichung der Lösung, ohne eine konkrete Implementierung zu besprechen oder vorzunehmen.	Beschreibung, wie ein Problem angegangen werden muss und Planung, welche Schritte zum Erfolg führen, z. B. den groben Algorithmus: „der Roboter muss immer nach links gucken, ob die Kiste noch da ist.“
Implementierungsphase (I)	Konkrete Interaktion mit der Programmierumgebung oder mit dem Roboter. Generierung oder Veränderung von konkretem Code oder der Konstruktion des Geräts.	Anschließen von Inputs und Outputs; Programmieren in der Programmierumgebung, Veränderung von Schleifenbedingungen und Werten, Einfügen von Blöcken.
Durchführungsphase (D)	Die Ausführung des Programms oder Teilen des Programms, sowie das Nutzen von Simulations- und Testumgebungen. Jeder Programmstart in einer der Umgebungen wird als eine Phase gezählt. Die SuS beobachten den Ablauf des Programms bis es beendet ist, oder sie brechen es ab.	Starten des Programms auf dem Gerät/Simulation/ Experimentierumgebung.

Auswertungsphase (A)	Direkte Reaktion der SuS auf die Durchführungsphase (gleichzeitig oder im Anschluss, ggf. wird Beobachtetes erst mehrere Phasendurchläufe später aufgegriffen und ausgewertet). Konkrete Vorschläge für vorzunehmende Veränderungen bezogen auf die drei vorhergehenden Phasen. Wird eine neue Hypothese formuliert oder ein Problem in Teilprobleme aufgegliedert, so führt dies direkt zu einer neuen Vorbereitungsphase.	Vorschlag für Veränderungen von Variablen, der Konstruktion, Startposition oder der Anfangshypothese; Beschreibung, Interpretation und Evaluation von beobachtetem Roboterverhalten: „der Roboter macht keine 90-Grad-Drehung so wie in unserem Programm“.

Tabelle 3.5: Codiermanual zur Forschungsfrage 1 (Analyse des Physical-Computing-Prozesses)

Name	**Indikatoren**	**Beispiel**	**Hinweis**
Vorbereitungsphase			
Ziel formulieren			
/Klären Aufgabenstellung	Lesen Aufgabenstellung; erarbeiten gemeinsames Verständnis für das Ziel der Aufgabenstellung	A: Was heißt der Fotowiderstand ist senkrecht aufgerichtet?	Teilweise durch Lesen der Aufgabenstellung und Verständnisfragen dazu ausgelöst
/Definieren vorliegendes Problem	Beschreiben wie ein Zielzustand erreicht werden kann; beschreiben Systemzustände, die Teil des Problems sind	B: Also wir haben drei Zustände. Wir haben halb-hell, dunkel und ganz-hell. Oder?	
/Identifizieren Problem	Sagen, dass es ein Problem gibt; beschreiben ein konkretes Problem	D: Er [der Roboter] kann nicht um die Seite fahren. Bevor... oder? E: Na wenn er nah genug dran ist, dass er es erkennt. D: Na da muss er schon richtig nah dran sein.	
/Holen Informationen ein	Erfragen Informationen bei der Lehrkraft, um die Aufgabenstellung zu verstehen oder einen Lösungsweg entwickeln zu können	A: Da ist unser Fotowiderstand. B: Wie funktioniert der? [...] B: Und misst der logarithmisch von der Skala her oder nicht logarithmisch?	

/Beschreiben Algorithmus	Beschreiben ein grobes Vorgehen, mit dem das Problem gelöst werden kann; geben einzelne Schritte des Vorgehens an und verbalisieren teilweise Kontrollstrukturen	B: Also beim Ultraschallsensor, wenn er darum fahren soll, dann muss man ihm sagen, er soll sich vor dem „in Acht nehmen“ was am nächsten dran ist. Denn es sind ja keine 2 m zwischen der Kiste und dem (Schrank). Ja, das gibts bestimmt ja als Ding [Block in Programm]. B: Da muss man ihm sagen, dass [grübelt] und dann müssen wir ihm sagen, dass er sich auch nicht mehr als 10 cm davon entfernen sollte, dass er halt darum fährt.	
/Formulieren Hypothesen	Beschreiben von Abhängigkeiten von mindestens zwei Variablen; nutzen „wenn, dann“-Konstruktionen in ihrer Beschreibung	B: Okay wir können erstmal anfangen, dass er losfährt, mit Distanz, dass wenn er die Kiste nicht mehr hat (sieht), dann soll er einfach stehen bleiben. Also soll er sich erstmal nicht drehen und nur stehen bleiben.; B: Also wenn er die Kiste nicht sieht, dann soll er sich immer so ein Stückchen drehen.	Die Hypothese bezieht sich zumeist auf das Verhalten/den Output des PhC-Geräts, um das ursprüngliche Problem zu lösen
Versuch planen			
/Planen Form des Experiments	Wählen die reale Erprobung aus; wählen die virtuelle Experimentierumgebung aus	E: Lass uns mal grundlegend gucken ob es richtig ist und mach alles nochmal weg. D: Warte, wir gehen mal hier in Projekt, neues Experiment. Wir gucken mal ob das überhaupt funktioniert	

/Planen das Vorgehen	Beschreiben den Lösungsansatz bzw. Algorithmus detaillierter, bspw. mit konkreten Werten und Zuständen; beschreiben benötigte Bauelemente oder Programmbestandteile, die für den Versuch benötigt werden.	B: Ja genau. Halb-hell soll also sein gelb. (...) Dann müssen wir dafür irgendwie Values setzen.	Planen auf Ebene des übergeordneten Problems/Ziels
Probleme antizipieren			
/Decken potentielle Schwierigkeiten/Problemursachen auf	Beschreiben allgemeine Probleme, die das Erreichen des Ziels erschweren oder verhindern können	B: Müssen wir den Resistor einbauen für die Lampen. Weil die Dioden leuchten ja heller wenn man da keinen Resistor mit reinbaut. Und dann ist es irgendwie besser zu sehen.	Verbleibt auf einer allgemeinen Ebene

Implementationsphase		
Auswahl von Input, Output und Verarbeitungsgeräten		
/Stellen Geräte zusammen	Wählen Inputs und Outputs aus, die sie für ihren Lösungsansatz benötigen; beziehen Verarbeitungsgerät als Gerät mit ein	A: Ich mach erst mal die Lampen rein. Rot grün und gelb, oder? B: Ja. A: Müssen da Widerstände mit ran? B: 10 $k\Omega$ glaube ich. Da stand jedenfalls 10 $k\Omega$. A: Der ist für den Fotosensor. Nicht für die Lampen.; B: Nimm mal längere Kabel.
/Nehmen Konstruktionen am Gerät vor	Bauen Konstruktion aus Input, Output und Verarbeitungsgerät zusammen; nehmen bauliche Veränderungen an der Hardware vor; Positionierung von Sensoren verändert	[L verändert die Kabelführung, da ein Rad gerade über das Kabel gefahren ist]; [bauen beide]
/Konstruieren Experiment	Nutzen Experimentierumgebung zum Testen von Sensoren und Aktuatoren; testen Sensoren und Aktuatoren direkt durch das Ausführen eines Programms auf dem Roboter	D: Okay, lass uns nochmal den Sensor testen. E: Wieso? [D testet Sensor mit Experiment] E: Lass uns mal schauen wo 50 cm für ihn sind.
/Prüfen Plausibilität des Vorgehens	Beschreiben erwarteten Output ihres Programms; gehen den Algorithmus durch und bewerten ggf. die Passung des Algorithmus zu dem gesetzten Ziel	E: Warte, lass uns erstmal gucken, was theoretisch passieren müsste. Er erkennt was in 50 cm, dann dreht er sich, fährt. [überlegen] D: Warte. Wenn er nichts erkennt, dann muss er sich drehen.

/Planen das Programm	Beschreiben konkrete Kontrollstrukturen, die sie in dem Programm verwenden wollen	D: Auf jeden Fall ist die Richtung hier verkehrt. Hier 10, da -10. E: Das 10 und -10 hebt sich auf. Weil erst fährt er nach links, rückwärts und dann nach rechts. Das bringt gar nichts. D: [verwirrt] Aber die Sache ist doch generell komisch. Er fährt nach links ... E: Er ist einfach zu nah dran, wenn er rückwärts fährt hat er noch Zeit zum drehen. D: Aber die Entfernung ist doch schon groß genug. Der ist doch nicht so nah dran. Theoretisch.	Zumeist parallel zum Programmieren; Planung auf der Ebene des Programms
Programmieren			
/Programmieren PhC-Gerät	Schreiben Programmcode in der Programmierumgebung bzw. stellen Programmblöcke per Drag & Drop zusammen	[programmieren]; E: Wir machen das jetzt mal komplett ohne Schleifen.	Sofern die SuS unverzüglich ihr Vorgehen beschreiben und umsetzen, ist es nicht der Planungsphase zuzuordnen.
/Überprüfen Programm	Gehen Programmcode schrittweise durch und entscheiden über die Passung zur Intention	[E überprüft dann Programm vor der Durchführung] E: Hä, da macht du gar keine Umdrehung rein! [Ja-Zweig] Der soll sich ja nicht drehen. D: Achso. E: Deswegen hört er auch nie auf.	

Durchführungsphase			
Programm ausführen			
/Wählen der Startposition des PhC-Geräts	Platzierung des Roboters vor der Kiste vor dem Programmstart; Diskussion über die für das Programm richtige Position des Roboters und anschließende Erprobung; Auswahl eines physischen Standorts für das PhC-Gerät	[Stellt Roboter erneut schräg zum Starten]; [probieren weitere Positionen]; D: Stell ihn mal näher ran. [an die Kiste]	Geschah bewusst und unbewusst; nur in Transkripten, in denen mit LEGO-Robotern gearbeitet wurde
/Starten des Programms	Drücken des Start-Buttons auf dem Roboter; drücken den Start-Button für den Roboter über die Programmierumgebung; drücken den Start-Button eines Experiments in der Programmierumgebung	[Starten das Programm]	Jeder Start wird einzeln codiert
/Brechen Programm ab	Beenden das Programm vorzeitig ab durch: Betätigung des Zurück-Buttons, Entfernung von Bauteilen oder das Umsetzen des Roboters	[Brechen Programm ab]; [probieren aus, Roboter dreht sich früher und in kleinen Schritten (nicht weit genug), weswegen B ihn nochmal etwas um die Kurve zieht. Durch Drall zur Kiste schafft Roboter es fast herum, bis er zu weit von der Kiste weggefahren ist]	Zumeist durch die transkribierende Person beschrieben

Beobachtung der Auswirkungen			
/Beobachten Programmausführung	Beobachten das Gerät bei der Ausführung des Programms; beschreiben die Auswirkungen des Programms	[probieren erneut 3 mal] E: Er macht eine Umdrehung, ne?; B: Dunkel ist 400. A: Ja, 400. B: So mittelhell ist. A: 700. B: 770. A: Ja. B: [leuchtet] und hell ist. Na, so (unverständlich 1780)?	Von der Transkribierenden Person festgestellt

Durchführungs- und Evaluationsphase		
Unsicherheiten und Fehler identifizieren		
/Erkennen Messunsicherheiten und Fehler	Erklären, dass Messunsicherheiten und Fehler in den Messungen mit Sensoren eine Rolle gespielt haben können; beschreiben vorliegenden Fehler, der die Messung beeinflusst haben kann	D: Vielleicht stimmt etwas mit dem Sensor nicht?
Evaluationsphase		
Vergleich der Auswirkungen mit Programmteilen		
/Halten Ergebnisse fest	Notieren sich in der Messung erhobene Werte; sagen, dass sie sich die Werte aufschreiben	B: Da. Dann müssen wir mal gucken, wenn es dunkel ist... Haben wir irgendwo Schmierpapier?
/Zuordnen von Programmblöcken zu Auswirkungen	Es wird besprochen welche Programmblöcke welche Reaktionen des PhC-Geräts hervorrufen; Output wird beschrieben und die Ursache auf der Seite des Inputs genannt	A: Ich dachte eigentlich, dass er sich nach links dreht und nicht nach rechts. D: Vielleicht haben wir auch irgendetwas zu viel drin [im Programm] .

/Bewerten verwendete Programmbausteine	Beschreiben welche Beobachtungen mit einem konkreten Programmbestandteil zusammenhängen; beurteilen, ob der Programmbestandteil geeignet ist bzw. erfolgreiche Auswirkungen in Bezug auf die Lösung zeigte	E: Wir machen das jetzt mal komplett ohne Schleifen. (...) D: Also doch eine Schleife.
/Wählen Veränderungen am Programm/Aufbau	Treffen Entscheidungen bzw. machen Vorschläge, welche Veränderungen am Programm/der Konstruktion des Geräts/dem Aufbau des Versuchs vorgenommen werden sollen; benennen konkrete Veränderungen von Werten/Positionen	D: Du musst ihn vielleicht nah an ein Objekt heran stellen. Wir haben ja 30 cm.
/Erklären Problem als gelöst	Beschreiben, dass das Programm den angestrebten Output erzeugt; erklären, dass der Zielzustand erreicht wurde	[Motor dreht sich] B: Aber dann setzt es angle noch mal auf 0. A: Ja eben. B: Also wenn wir jetzt da eine Lampe drauf scheinen, dann sollte es zwar direkt abbrechen, aber auch wieder auf 0 setzen. B: Hey, es funktioniert oder? A: Mach mal noch mal. Und halt mal. Ja es funktioniert. L: Super. B: Also es funktioniert. Es stoppt wenn es eine Lampe sieht.

Identifikation des Problems		
/Suchen nach Problemursachen	Besprechen geschriebenes Programm; überprüfen die Konstruktion des Geräts; interagieren mit dem Gerät zur Laufzeit des Programms	A: Wir können ja mal die Werte sehen die es ausliest bei normal. Das ist gerade irgendwie gar nicht das. Ich kann ja mal gucken wie ich es angeschlossen habe; [prüft die Verkabelung]
/Erkennen mögliche Problemursachen	Nennen konkrete Ursachen, die sie in Zusammenhang mit ihrem Problem sehen; die Ursachen beziehen sich auf das PhC-System	D: Okay, mache mal deine Hand davor [vor den Sensor]. Näher, näher. Ja. E: Also hier ist erstmal der Roboter [Kante vorne] D: Ca. 10. Nein. Mach mal deine Finger zusammen [vor dem Sensor]. Ja, das ist so bei 18 cm. Also erkennen tut er was. Mach mal deine Hand ein bisschen höher, höher, höher... ja okay, das steigert sich nicht. [beide etwas ratlos]; D: Gucke mal, da gibt es zweimal den Ultraschallsensor, ach, ist ja Infrarot. E: Das ist der Ultraschall übrigens den wir benutzen müssen! D: Haben wir hier Ultraschall? Ja!

Entscheidung eine Variable zu ändern			
/Verändern Variable	Nehmen Veränderungen an einer konkreten Variable vor	[probiert an der Kiste, Roboter fährt gegen die Kiste und hält dann direkt an] E: Er hat nur eine Umdrehung gemacht. D: Ja, weil er die Kiste erkannt hat. [E geht zurück an den Platz] E: Jetzt nimm mal das Erste raus [aus dem Programm] D: Also dass er nicht fährt? [D überlegt was dann passiert] [E probiert derzeit aus, Roboter macht 2 Radumdrehungen und bleibt vor der Kiste stehen] D: Es funktioniert.	
Ziel in Teilaufgaben gliedern			
/Formulieren Teilziele	Erklären, welche Schritte zur Problemlösung erreicht werden müssen; gehen bspw. auf fehlende Daten ein, die zunächst vorliegen müssen	E: Lass uns mal grundlegend gucken ob es richtig ist und mach alles nochmal weg. D: Warte, wir gehen mal hier in Projekt, neues Experiment. Wir gucken mal ob das überhaupt funktioniert. [lesen Sensorwerte des Roboters mit der Experimentierumgebung aus]	Das Auslesen von Sensorwerten ist dabei der Durchführungsphase zuzuordnen.
Entscheidung für erneuten Start der (Teil-)Schleife			
/Verändern ihre Hypothese	Revidieren ihre Hypothese und passen sie an gewonnene Daten an	E: Hä, da macht du gar keine Umdrehung rein! [Ja-Zweig] Der soll sich ja nicht drehen. D: Ach so. E: Deswegen hört er auch nie auf. [probieren aus, Roboter fährt geradeaus, hat Linksdrall und fährt deswegen an Kiste vorbei] D: Er dreht sich, warte wir haben etwas falsch gemacht. D: Jetzt müssen wir das so überarbeiten, dass wenn er etwas erkennt, das er sich dreht und weiter fährt.	

/Entscheiden über weiteres Vorgehen	Konkreter Entschluss über die nächste Aktion, um der Problemlösung näher zu kommen; gefolgt von dem Übergang in eine andere Hauptphase	B: Wir nehmen mal den (Kabel an Breadboard)... ach, nee. Den müssen wir ja gar nicht rausnehmen. Und das ist plus elf. Warte mal kurz. Wollen wir erst mal überhaupt schauen, ob ... A: Die Lampe angeht. B: Genau. Ergibt Sinn.	Abhängig von der Entscheidung wird in die Phasen Planung, Implementation oder Durchführung übergegangen

Phasenunabhängiges Vorkommen		
Sonstiges	Off-topic-Gespräche,	F: Ein bisschen wie Nicky, als ob es die seltsamste Variante überhaupt wäre. [Gespräch über eine dritten Person]; B: Ich kann mit Kulli einfach überhaupt nicht schreiben.
	Verbale Äußerung oder Gestik lässt sich keiner der Kategorien zuordnen	
Umgang mit Problemen	Probleme werden benannt; Probleme werden beschrieben; Nennung, das ein (unbekanntes) Problem vorliegt ohne konkrete Erklärung [stellt Roboter erneut schräg zum starten]	A: Wenn er hier abbiegt, dann sieht er die Kiste ja nicht mehr und fährt einfach weiter. [alte Idee des Vierecks]

/Ziehen Vorwissen heran	Nutzen bereits vorhandene Kenntnisse über Programmierung oder Robotik; nutzen Wissen, dass sie im Rahmen des Gesamtprozesses zur Aufgabenlösung erworben haben	A: Ähm. Wie lesen wir den Sensor aus? Die Theorie ist, dass wir... Wo haben wir den Sensor? Hast du den Sensor? B: Hier oder? A: Ja. Wunderbar. Dass wir den quasi als Widerstand schalten. Wir brauchen einen kleinen extra Widerstand. Nur damit wir hier keinen Kurzschluss verursachen. L: Nur den 10 $k\Omega$. A: Nur den 10 $k\Omega$. Ok gut. A: 10 $k\Omega$. B: [guckt auf den Aufgabenzettel] A: Das hatte sie am Anfangs gesagt. B: 10 $k\Omega$ sind braun-schwarz-schwarz. L: Das steht auch an der Tafel im Zweifelsfall. A: Ok. Das ist auch gut. Also braun-schwarz-schwarz-rot-braun? B: Wollen wir mal das Projekt von gestern aufrufen [am PC], da war es ja so ähnlich. Vielleicht können wir es ja übernehmen nur mit anderen Werten. [überlegen] Okay, so ähnlich haben wir es eigentlich. [überlegen, klicken mögliche Optionen im Code durch]	Kann Kommentare der Lehrkraft beinhalten, wenn die Lehrkraft aus eigener Initiative eingreift. Diese Bemerkungen sind jedoch nicht ausschlaggebend für die Verwendung des Codes.
/Nutzen Informationen der Lehrkraft	Beziehen Informationen ein, die sie von der Lehrkraft bekommen haben; Informationen, die sie auf Rückfrage bekommen haben oder Informationen aus der Aufgabenstellung	L: Also das 5 V (-Kabel) kommt auf jeden Fall auf die eine Seite vom Sensor. B: Ach so. Ok. Dann A0. Was ist denn der mittlere? L: Die Daten. B: Die Daten. Ach so. [baut um] B: So. Daten. Und jetzt wo ist der Widerstand?	

Geräte und Software I

Alle Studien werden mit Physical-Computing-Geräten durchgeführt, deren Eigenschaften in diesem Abschnitt kurz zusammengefasst werden. Da ebenfalls eine große Vielfalt an Software für die Programmierung dieser Geräte existiert, wird auf die hier verwendete Software eingegangen. Alle Studien werden mit LEGO Mindstorms-Robotern oder Arduino-Mikrocontrollern durchgeführt.

Die verwendeten LEGO Mindstorms-Roboter (Generation EV3) werden seit 2013 von LEGO vertrieben und sind seitdem ebenfalls an die Schulen gelangt. Als neueste Version der LEGO Mindstorms-Roboter ähneln sie der vorherigen NXT-Generation stark und stellen somit keine großen Hürden für diejenigen dar, die bereits mit NXT-Robotern gearbeitet haben. Die Standardbauweise umfasst zwei Motoren, die jeweils ein Rad ansteuern. Darüber hinaus sind zwei Berührungssensoren (oder auch Drucksensoren, Tastsensoren) an jeweils einer Vorderseite angebracht. Ebenfalls in Fahrtrichtung ausgerichtet befindet sich ein Ultraschallsensor, der circa 20 cm über Bodenhöhe und 15 cm hinter den Berührungssensoren angebracht und dadurch schräg hinter dem Brick (der Steuereinheit) positioniert ist (siehe Abbildung 3.2). Darüber hinaus umfasst der Bausatz einen Gyrosensor (Lagesensor), einen Lichtsensor und einen weiteren Motor (LEGO GmbH, 2017). Diese zuletzt genannten Elemente werden in den Studienaufgaben nicht verwendet.

Der Roboter kann über eine blockbasierte Programmiersprache namens *Open Roberta*[3]

Abbildung 3.2: LEGO Mindstorms-Roboter, verwendete Standardbauweise

(siehe Abbildung 3.3, Version 2.2.7) angesteuert werden. Sie ist webbasiert und kann zur Programmierung weiterer Physical-Computing-Geräte genutzt werden. Ein großer Vorteil besteht darin, dass sie der Programmiersprache *Scratch*[4] ähnelt und dadurch für die SuS bekannt und leicht verständlich sein sollte. In Open Roberta ist es möglich, das geschriebene Programm auf einem virtuellen Roboter zu simulieren und z. B. im Anschluss auf einem LEGO Mindstorms-Roboter zu testen. Die SuS werden instruiert die Simulationsfunktion nicht zu nutzen, um die Vergleichbarkeit der einzelnen Studien zu wahren. Des Weiteren würde eine Simulation den Studienzielen widersprechen, die eine Interaktion mit der physischen Umgebung verlangen.
Der EV3-Roboter wird mit einer Software von LEGO ausgeliefert (LEGO Mindstorms Education EV3 Software[5] siehe Abbildung 3.4), die für die 1. - 3. Studie verwendet wird. Diese ist ebenfalls blockbasiert, jedoch in der Handhabung und im grafischen Aufbau nicht mit geläufigen blockbasierten Programmierumgebungen vergleichbar. Zum Beispiel wird das Programm horizontal aufgebaut, wohingegen Programme wie Scratch vertikal aufgebaut werden. Das hat u. a. den Vorteil, dass ohne Scrollen mehr Teile des Programms auf dem Bildschirm sichtbar sind. Einzelne Einstellungen, die am Programmblock vorgenommen werden können, sind in der Programmierumgebung von LEGO durch Symbole realisiert. Scratch ist an dieser Stelle hingegen eher textlastig, somit aber ebenfalls mit weniger Interpretationsspielraum behaftet.

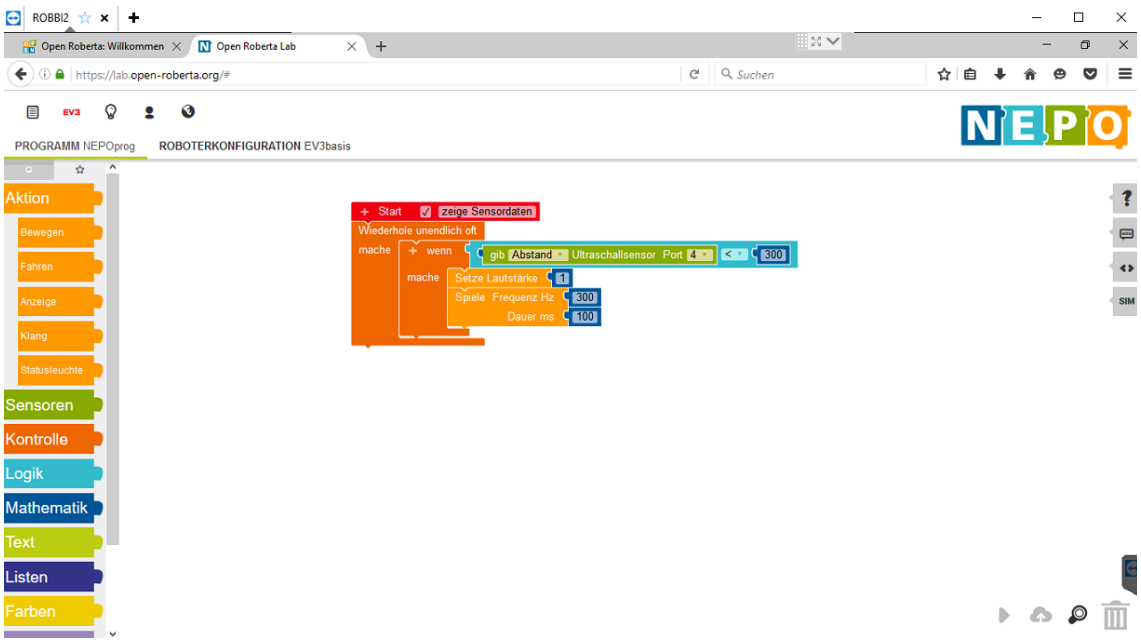


Abbildung 3.3: Programmierumgebung Open Roberta

[3] https://lab.open-roberta.org
[4] https://scratch.mit.edu
[5] https://education.lego.com/de-de/downloads/mindstorms-ev3

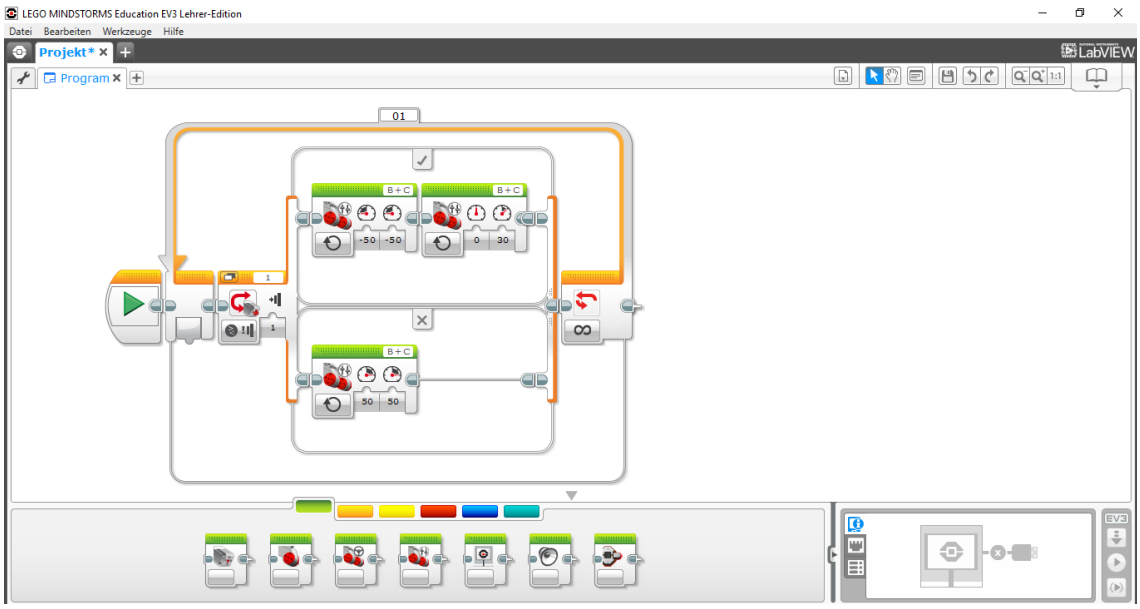


Abbildung 3.4: Programmierumgebung von LEGO Mindstorms

Des Weiteren stellt die Programmierumgebung von LEGO Mindstorms eine Experimentierumgebung zur Verfügung, bei der Sensorwerte direkt ausgelesen werden können und ein entsprechender grafischer Verlauf angezeigt wird (siehe Abbildung 3.5). Diese lässt Experimente mit Hilfe der Sensoren zu, ohne dass die Sensoren in einem Programm abgefragt werden müssen. Die Sensorfunktionen können dadurch isoliert betrachtet werden, so dass zum Beispiel Sensorgrenzen ermittelt werden können.

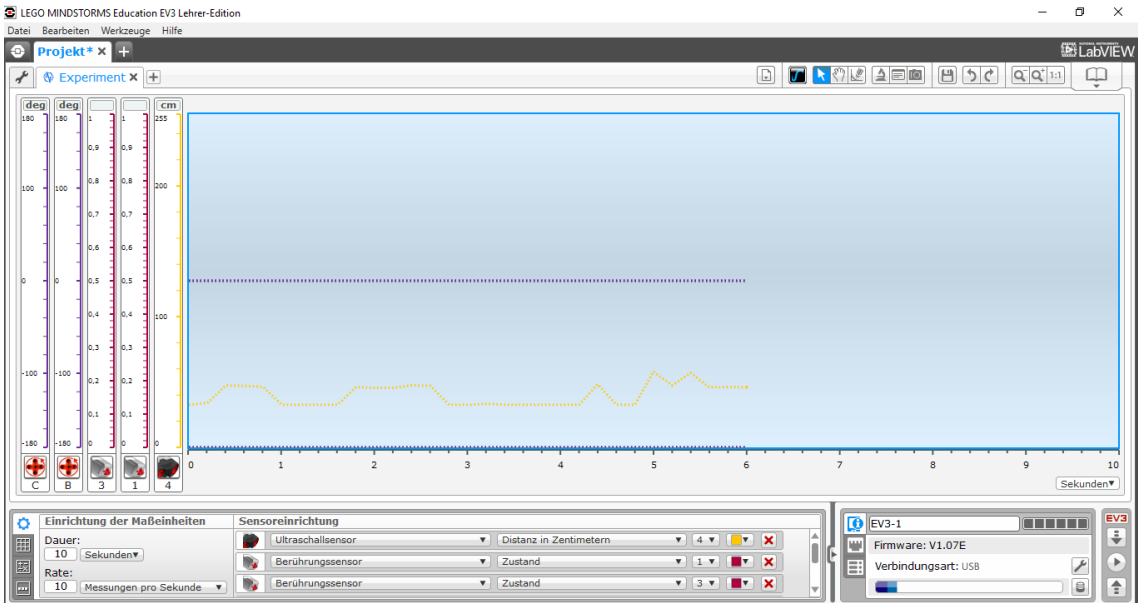


Abbildung 3.5: Experimentierumgebung von LEGO Mindstorms

Arduino-Mikrocontroller der Generation Uno R3 (siehe Abbildung 3.6, Arduino) werden in einer weiteren Studie verwendet. Dafür wird ein Satz von Sensoren und Ak-

tuatoren von Sunfounder[6] genutzt. Das Arduino Board ist ausgestattet mit 14 digitalen Input/Output-Pins und 6 analogen Input-Pins. Es arbeitet mit bis zu 5-Volt-Output und 7 – 12-Volt-Input (Arduino, 2017). Zusätzlich zu dem Arduino werden handelsübliche Breadboards und Jumper Wires zur Verfügung gestellt (in dem Kit von Sunfounder enthalten), um weitere Peripherie an den Arduino anzuschließen. Dabei werden für Übungsaufgaben und die Studienaufgaben (vgl. Abschnitt 1.3.3) folgende Materialien genutzt: diverse Widerstände, verschiedenfarbige LED-Lampen, RGB-LED-Lampen, Servomotoren, analoge Lichtsensoren, analoge Temperatursensoren, Potentiometer, Drucksensoren (enthalten im Sunfounder-Starter-Kit).

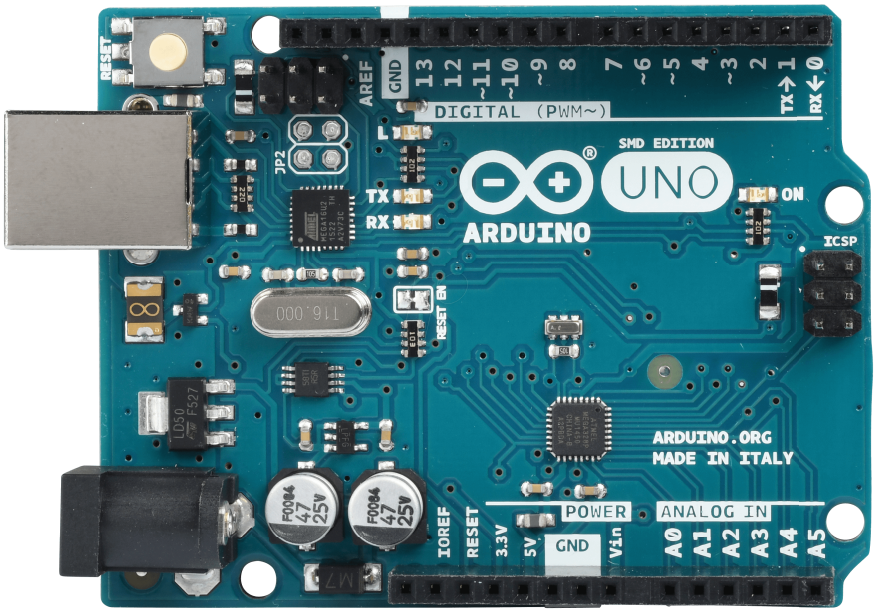


Abbildung 3.6: Arduino Uno R3-Mikrocontroller, ohne Peripherie

Zur Ansteuerung des Arduino-Mikrocontrollers wird die Software *Scratch for Arduino*[7] (siehe Abbildung 3.7, Version 1.6) verwendet. Scratch for Arduino (S4A) ist mit Scratch weitgehend identisch, wobei es zusätzlich möglich ist einen Arduino zu programmieren und direkt in der Umgebung Sensorwerte auszulesen. Die sogenannte *Bühne* (auf der das geschriebene Programm simuliert ausgeführt wird) ist an der gleichen Position wie bei Scratch, jedoch handelt es sich mit dem Arduino-Mikrocontroller um eine physische Komponente und keine Simulation. Des Weiteren wurden einige Blöcke (auf der linken Seite) angepasst oder sind neu hinzugekommen, da sie für die Ansteuerung des Arduino-Mikrocontrollers benötigt werden. Das sind z. B. Blöcke, die analoge und digitale Inputs/Outputs ansteuern. Ein Nachteil für eine weiterführende Arbeit mit dieser Software besteht darin, dass das Programm nicht auf dem Mikrocontroller gespeichert werden kann. Somit wird für die Ausführung des Programms ein weiterer Computer benötigt.

[6] https://www.sunfounder.com/starterkit/arduino/super-kit-v2-0/super-kit-v2-0-for-arduino.html
[7] http://s4a.cat

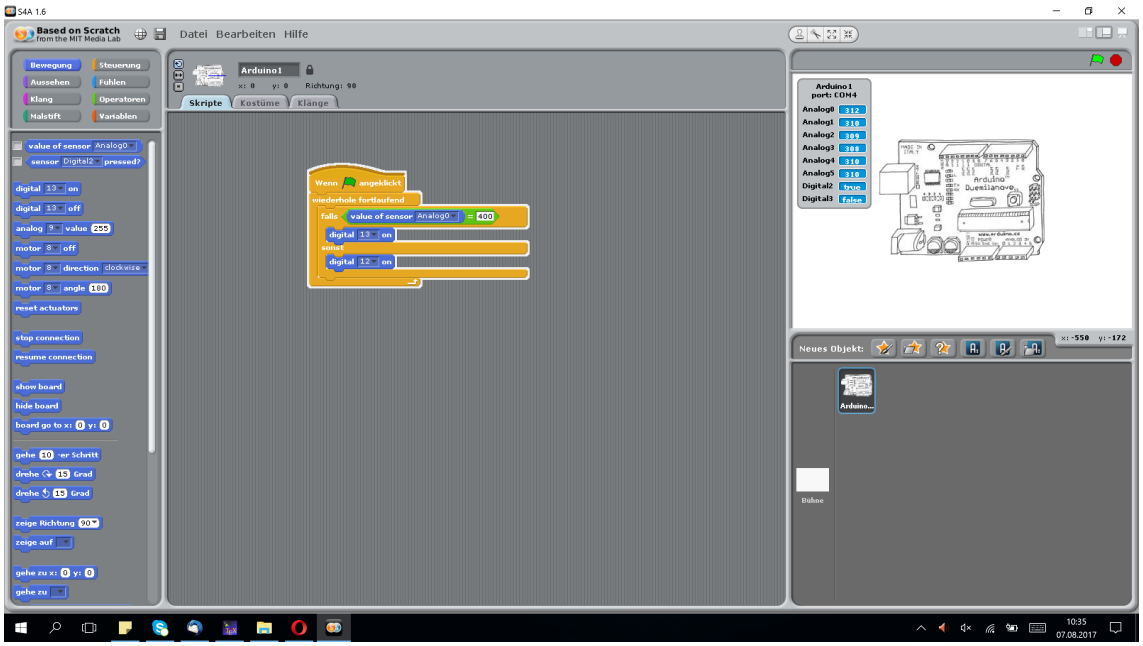


Abbildung 3.7: Programmierumgebung Scratch for Arduino

3.2.3 Ablauf der Datenerhebung I

Während der Datenerhebung war nur die Versuchsleitung im Raum anwesend, um die Aufgaben zu stellten und die Kamera(s) zu bediente. In den Studien 1 - 4 sollten möglichst keine Fragen gestellt werden. Wenn den SuS jedoch Informationen fehlten (wie z. B. ob ein Sensor analog oder digital misst), dann wurden diese Fragen beantwortet. Hatte die Lehrkraft den Eindruck, dass die SuS demotiviert waren (durch Körperhaltung oder verbale Äußerungen) und nicht sicher war, ob sie die Aufgabe fertig stellen konnten, griff die Lehrkraft ein und gab den SuS Hinweise. Dies geschah insbesondere kurz vor dem Ende der Bearbeitungszeit, damit die SuS die Aufgabe nach Möglichkeit fertig stellen konnten. Lediglich bei der 1. Studie waren beide Gruppen gleichzeitig in dem Raum und eine weitere Lehrkraft stand zur Verfügung.
In den vier Studien waren ein bis zwei Kameras im Raum, die das Geschehen aufzeichneten sowie gerichtete Mikrofone, um das Gesprochene im Anschluss besser transkribieren zu können. Die Kameras filmten in den Studien über die Schultern der ProbandInnen, so dass der Programmcode auf den Computern und die Bewegungen des Roboters bei der Ausführung zu sehen waren. In der 4. Studie war zusätzlich eine Kamera auf den Arduino-Mikrocontroller gerichtet, damit genau untersucht werden konnte, welche Änderungen bei der Konstruktion vorgenommen wurden.

3.3 Empirische Untersuchung des Physical-Computing-Prozesses

Die theoretischen Gemeinsamkeiten und Unterschiede des Physical-Computing-Prozesses und von dem Prozess der wissenschaftlichen Erkenntnisgewinnung wurden bereits im ersten Teil dieses Kapitels (Kapitel 3.1) dargelegt. In diesem Abschnitt wird der zweite Teil der Forschungsfrage (*Kann Physical Computing als Arbeitsweise der Erkenntnisgewinnung in der Informatik genutzt werden?*) beleuchtet. Dies geschieht durch ein empirisches Fundament und einer Adaption des theoretischen Physical-Computing-Modells. Es soll ermittelt werden, ob das Phasenmodell des Physical Computing mit Hilfe empirischer Daten gestützt werden kann. Davon ausgehend soll die Reihenfolge der Phasen im Physical-Computing-Prozess beschrieben werden. Dafür werden Methoden und Instrumente genutzt, die in der Phase Design & Construction I des DBR-Prozesses zur Beantwortung der Forschungsfrage zusammengestellt wurden. An dieser Stelle wird in die Phase Evaluation & Reflection I übergegangen.
Eine Motivation für diese Untersuchung ist, dass in der existierenden Literatur bislang nicht ersichtlich ist, ob die beschriebenen Phasen des Physical Computing aus empirischen Daten oder normativ abgeleitet wurden. Durch die Beantwortung der Frage, welche Ähnlichkeit beide Prozesse aufweisen, kann die Entscheidung über den Nutzen von Physical Computing als Arbeitsweise der Erkenntnisgewinnung getroffen werden.

Die Datenbasis besteht aus Transkripten der ersten vier Studien, an denen insgesamt 25 SuS teilgenommen haben (vgl. Kapitel 1.3.2). Diese teilen sich in 13 Gruppen auf, wobei 12 davon paarweise arbeiteten und eine Person alleine arbeitete. Somit liegen 13 Transkripte zugrunde, die qualitativ untersucht wurden. Innerhalb einer Studie werden die Gruppenmitglieder durch Buchstaben bezeichnet. Dadurch existieren für jede Teilstudie die Gruppenpartner *AB*, *CD* usw. Ist nur ein Buchstabe angegeben, handelt es sich um die Person, die alleine arbeitete. Als Präfix wird die Studiennummer als *S1* bis *S4* angegeben. Die Anzahl aller Codings über alle Transkripte hinweg beträgt $N = 869$. Die Phase der Vorbereitung umfasst $N = 51$, die Implementationsphase $N = 261$, die Durchführungsphase $N = 316$ und die Evaluations- bzw. Auswertungsphase $N = 241$. Die Anzahl von Codings innerhalb der Transkripte schwankt zwischen 24 und 181, wobei der Mittelwert bei $MW = 67$ liegt. Eine genaue Aufgliederung der Verteilung von Hauptphasen auf die unterschiedlichen Gruppen kann den Abbildungen 3.8 und 3.9 entnommen werden. In Abbildung 3.8 wird zunächst die numerische Verteilung von Phasen auf die einzelnen Transkripte dargestellt und aufbauend darauf sind die Zahlen in Abbildung 3.9 grafisch durch Quadrate repräsentiert. Die Fläche der Quadrate steht in Relation zu der Codehäufigkeit und gibt deren Gewichtung an. Es wird eine Code-Matrix erzeugt, um die Daten zu visualisieren. Dafür wird eine Funktion von MaxQDA genutzt, welche die Fläche auf einer siebenstufigen Skala abhängig vom Maximum berechnet. Anhand der grafischen

Darstellung ist zu erkennen, dass es sich bei der Gruppe DE der 2. Studie um eine herausragende Gruppe handelt, die besonders viele Implementations-, Durchführungs- und Evaluationsphasen im Vergleich zu den anderen 12 Gruppen durchlief. Alle Gruppen waren zwischen ein- und achtmal in der Vorbereitungsphase, welche bei allen die geringste Anzahl von Phasen einnimmt. Durchschnittlich hielt sich jede Gruppe viermal in dieser Phase auf. Die Implementationsphase wurde acht- bis 53 mal mit einem $MW = 20$ durchlaufen. In der Phase der Durchführung hielten sich alle ProbandInnen zwischen sieben- und 74 mal mit einem $MW = 24$ auf. Für die Auswertungsphase wurde eine Anzahl von fünf- bis 48 und einem $MW = 19$ ermittelt. Damit handelt es sich bei der Vorbereitungsphase um die mit der vergleichsweise deutlich geringsten Anzahl von Phasen. Die Anzahl der anschließenden Implementationsphasen verdoppelt bis vervielfacht sich, wobei die Anzahl der Durchführungsphasen ungefähr gleich bzw. teilweise höher als die Implementationsphase ist. In ihrer Anzahl nimmt die Auswertungsphase jedoch – teilweise rapide – ab.

Codesystem	S1-AB	S1-CD	S2-AB	S2-C	S2-DE	S3-AB	S3-CD	S3-EF	S4-AB	S4-CD	S4-EF	S4-IJ	S4-KL
Vorbereitung	4	6	6	3	6	4	2	3	8	2	4	2	1
Implementation	21	14	23	22	53	13	23	21	25	13	8	9	16
Durchführung	24	15	29	28	74	16	29	28	27	13	7	9	17
Auswertung	15	13	25	21	48	17	24	21	22	9	5	8	13

Abbildung 3.8: Numerische Darstellung der Code-Matrix für alle analysierten Transkripte

Abbildung 3.9: Graphische Darstellung der Code-Matrix für alle analysierten Transkripte

Als weitere Form der Phasenanalyse steht eine Funktion von MAXQDA zur Verfügung, die Beziehungen von Codes zueinander ermittelt, die sogenannte Code-Relation. Alle Transkripte werden dabei einzeln betrachtet und Überschneidungen bzw. die Nähe von Codes wird gezählt. Bei den Überschneidungen handelt es sich um *echte* Überschneidungen von Codes. Eine Nähe beschreibt, welche Codes aufeinander folgen, z. B. auf Basis von Zeilenabständen. Für die folgenden Analysen wurde das Maß der Code-Nähe mit einem Zeilenabstand von $N = 1$ genutzt, d. h. die Codes grenzen direkt aneinander. Dieses Vorgehen ist angemessen, da die untersuchten Phasen klar voneinander abgegrenzt werden und somit zeilenweise kaum Überschneidungen vorliegen (vgl. Beispiel 3.3.3). Nur in wenigen Fällen waren die Phasen sehr kurz und eine Zeile konnte mit mehreren Phasen codiert werden. Da die Relation der vier Phasen zueinander ermittelt wurde, zeigt sich in der Abbildung eine Symmetrie in der Diagonale. In den Abbildungen 3.10 und 3.11 ist numerisch und grafisch dargestellt wie häufig unterschiedliche Phasen aneinander angrenzen (bzw. eine Nähe ausweisen) und dadurch eine Relation zwischen ihnen besteht. Die stärkste Relation existiert zwischen der Phase der Implementation und der Durchführung mit einem Wert von $N = 487$ für eine Nähe der Codelines mit $N = 1$. Das heißt, dass die Phasen Implementation und Durchführung genau 487 mal direkt aneinander angrenzten. Eine häufige Nähe ist ebenfalls bei der Phase der Durchführung und Auswertung deutlich zu erkennen mit einem Wert von $N = 482$. Die Phasen der Implementation und Auswertung weisen häufig eine Nähe auf mit einem Wert von $N = 331$. Die Phasen der Vorbereitung und die Implementation weisen teilweise eine Nähe auf mit $N = 106$, wobei die Vorbereitungsphase und die Durchführung sowie Auswertung selten in der Nähe voneinander sind ($N = 16$ bzw. $N = 33$). Das liegt unter anderem darin begründet, dass es nur wenige Vorbereitungsphasen im Vergleich zu den anderen Phasen gab. Die Vorbereitungsphase nimmt einen prozentualen Anteil von 5,9 % aller Codings ein. Es wurden keine Abschnitte in den Transkripten gefunden, die nicht einer der vier Phasen zugeordnet werden konnten. Mit Hilfe eines zweiten Codierers wurde die Interrater-Reliabilität Cohens Kappa bestimmt. Es wurden 10 % der Transkripte von der 1. und 2. Studie untersucht (daher circa 5 % aller Transkripte für diese Forschungsfrage). Für die Vorbereitungsphase wurde eine Übereinstimmung von $\kappa = 0.68$ ermittelt. Es wurden Werte von $\kappa = 0.74$ für die Implementationsphase, $\kappa = 0.86$ für die Durchführungsphase und $\kappa = 0.68$ für die Auswertungsphase berechnet. Verglichen mit Landis und Koch zeigt sich eine substantielle bis annähernd perfekte Übereinstimmung beider Codierer.

Bis hierhin wurde die Gesamtzusammensetzung der Phasen über alle Studien dargestellt. Im Folgenden werden die einzelnen Studien in Hinblick auf verwendete Gerätetypen (LEGO-Roboter oder Arduino-Mikrocontroller) unterteilt und getrennt voneinander betrachtet. Diese sind ebenfalls grafisch und numerisch in den Abbildungen 3.14, 3.15, 3.16 und 3.17 dargestellt. Es ist zu erkennen, dass sich die numerischen Werte zwischen den Studien mit LEGO-Robotern und Arduino-Mikrocontrollern in ihrer Ausprägung unterschei-

Codesystem	Vorbereitung	Implementation	Durchführung	Auswertung
Vorbereitung		106	16	33
Implementation	106		487	331
Durchführung	16	487		482
Auswertung	33	331	482	

Abbildung 3.10: Numerische Code-Relation-Darstellung über alle Transkripte

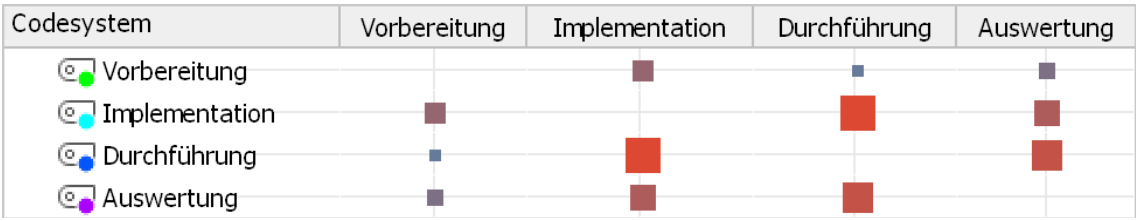


Abbildung 3.11: Grafische Code-Relation-Darstellung über alle Transkripte

den. Das liegt darin begründet, dass weniger Studien mit Arduino-Mikrocontrollern als mit LEGO Mindstorms-Robotern durchgeführt wurden. Bei der Analyse der Nähe von Phasen zeigt sich eine ähnliche Verteilung trotz unterschiedlich starker Ausprägung bei beiden Gerätetypen. In den Studien mit LEGO-Robotern ist die Nähe der Implementationsphase und der Durchführung schwächer ausgeprägt als die Nähe der Durchführungsphase und der Auswertung. Die Gruppe, die mit Arduino-Mikrocontrollern arbeitete, zeigt dabei ein umgekehrtes Bild, in dem die Implementationsphase mit der Durchführungsphase häufiger aneinander angrenzen als die Durchführungsphase mit der Auswertungsphase. Die Nähe von der Vorbereitungsphase und der Implementationsphase sind hingegen bei den Studien mit Arduino-Mikrocontrollern im Verhältnis stärker ausgeprägt, was in den LEGO Mindstorms-Gruppen schwächer zum Vorschein kam.
Da in der 3. Studie eine Hilfestellung (*Evaluationsphase digital*) gegeben wurde, die zum Ziel hatte, die Auswertungsphase zu intensivieren, wurden die Daten dieser Studie im Folgenden aus der Code-Relation-Darstellung entfernt. Abbildungen 3.12 und 3.13 zeigen dabei leichte Veränderungen in den zuvor beschriebenen Gewichtungen. Die Phasen der Vorbereitung und Implementation weisen eine häufigere Nähe in den Studien auf, in denen der Prozess der SuS nicht unterstützt (bzw. nur ein Stimulus durch Hilfestellung *4-Phasen analog* gegeben) wurde. Ohne die explizite Unterstützung der Auswertungsphase wiesen die Durchführungsphase und die Auswertung weniger häufig eine Nähe zueinander auf als die Implementationsphase und die Durchführung. Allerdings handelt es sich dabei um einen marginalen Unterschied in Relation zu der Anzahl aller Phasen. Dieses Ergebnis kann darauf zurück geführt werden, dass durch die Unterstützung die Auswertungsphasen häufiger stattfanden. Statistische Ungenauigkeiten sind jedoch ebenfalls naheliegend, da sich diese Relation auch in der Arduino-Studie zeigt (vgl. Abbildungen 3.13 und 3.17). In diesem Zusammenhang können keine Aussagen darüber getätigt werden, ob durch die

Hilfestellung ein Einfluss auf die Qualität der Evaluation bestand. Zusammengefasst fällt auf, dass sich die Gewichtung der Phasen sowie ihre Nähe zueinander innerhalb der beiden verglichenen Studien geringfügig unterscheidet. Die Verwendung dieser Hilfestellung (*Evaluationsphase digital*) scheint bei der quantitativen Betrachtung einen marginalen Einfluss auf den Prozess des Physical Computing gehabt zu haben.

Codesystem	Vorbereitung	Implementation	Durchführung	Auswertung
Vorbereitung		50	7	20
Implementation	50		260	160
Durchführung	7	260		255
Auswertung	20	160	255	

Abbildung 3.12: Numerische Code-Relation-Darstellung über alle LEGO Mindstorms-Roboter-Studien, ohne Unterstützung

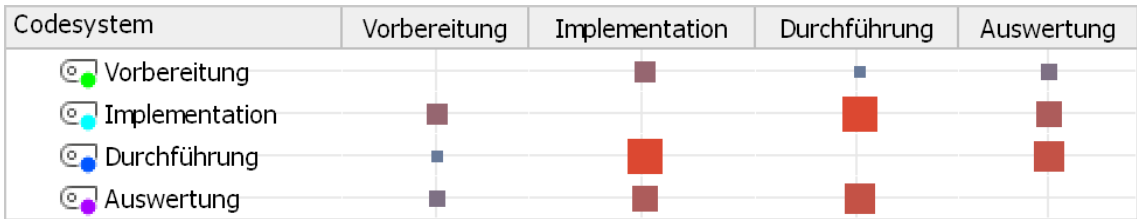


Abbildung 3.13: Graphische Code-Relation-Darstellung über alle LEGO Mindstorms-Roboter-Studien, ohne Unterstützung

Codesystem	Vorbereitung	Implementation	Durchführung	Auswertung
Vorbereitung		66	14	25
Implementation	66		362	249
Durchführung	14	362		368
Auswertung	25	249	368	

Abbildung 3.14: Numerische Code-Relation-Darstellung über alle LEGO Mindstorms-Roboter-Studien

Codesystem	Vorbereitung	Implementation	Durchführung	Auswertung
Vorbereitung				
Implementation				
Durchführung				
Auswertung				

Abbildung 3.15: Grafische Code-Relation-Darstellung über alle LEGO Mindstorms-Roboter-Studien

Codesystem	Vorbereitung	Implementation	Durchführung	Auswertung
Vorbereitung		40	2	8
Implementation	40		125	82
Durchführung	2	125		114
Auswertung	8	82	114	

Abbildung 3.16: Numerische Code-Relation-Darstellung über alle Arduino-Mikrocontroller-Studien

Codesystem	Vorbereitung	Implementation	Durchführung	Auswertung
Vorbereitung				
Implementation				
Durchführung				
Auswertung				

Abbildung 3.17: Grafische Code-Relation-Darstellung über alle Arduino-Mikrocontroller-Studien

3.3.1 Beschreibung der Teilphasen des Physical-Computing-Prozesses

Die hier vorliegenden Ergebnisse beziehen sich ebenfalls auf das bereits publizierte Paper von Schulz und Pinkwart (2016) und ergänzen darin dargestellte Daten. Bei der Codierung von den Transkripten wurde sich zunächst auf die Hauptphasen beschränkt, da diese für weiterführende Untersuchungen im Vordergrund stehen. Nichtsdestotrotz finden Teilphasen statt und wurden bereits theoriegeleitet beschrieben (vgl. Abbildung 3.1). Die empirische Überprüfung dieser Phasen wurde in dieser Untersuchung ebenfalls vorgenommen. Dafür wurden die, in der Physical-Computing-Literatur definierten, Teilphasen codiert und erweitert. Im Folgenden wird eine kurze Beschreibung der Teilphasen mit den dazugehörigen, beispielhaften Auszügen aus den Transkripten vorgestellt. Eine vollständige Auflistung der Transkriptionsregeln ist in den Tabellen E.1 und E.2 dargelegt. Runde Klammern beschreiben ausgelassene Worte, wenn ein Satz nicht beendet wurde, das Wort jedoch implizit durch den Kontext klar wird. Eckige Klammern beinhalten Anmerkungen der transkribierenden Person, beispielsweise über Handlungen der SuS oder eine Interpretation des Gesagten.

Ziel formulieren

Definition: Ist zumeist die erste Teilphase der Vorbereitungsphase, in der die Zielstellung geklärt wird und somit die zu erwartenden Auswirkungen des Programms. Es wird noch nicht detailliert darauf eingegangen wie dieser Zielzustand erreicht wird.

Beispiel 3.3.1 *3. Studie, Gruppe EF:*

E: *Wir stellen ihn davor, er sieht, dass da etwas ist. Und ich denke mal, dass es auch reicht wenn er in eine Richtung darum fährt. Also er fährt da hin, sieht, dass da etwas ist und kann sich ja erstmal nach rechts drehen oder so.*

Beispiel 3.3.2 *4. Studie Gruppe EF:*

F: *Also. Was genau müssen wir jetzt? Inwiefern sollen wir das suchen?*

E: *Wir sollen sozusagen Licht suchen. Das heißt, wir sollen uns in die Richtung bewegen, in der der Fotowiderstand, also dieses Ding, weniger anzeigt. Also niedriger wird. Weil er wird doch niedriger je heller es ist.*

Versuch planen

Definition: Beschreibt das gezielte Vorgehen, um das formulierte Ziel zu erreichen. Ein grundlegender Algorithmus wird beschrieben.

Beispiel 3.3.3 *4. Studie, Gruppe AB:*

A: *Wir brauchen sowieso erst mal einen Normalwert, damit wir uns an dem Wert orientieren können.*

B: *Ja klar. Also, halt, wenn der einen bestimmten Wert hat... Wollen wir das wirklich als If-Schleife machen? Oder wollen wir das mit dem solange-bis machen? Weißt du, wie ich meine?*

A: *Ja, ja.*

B: *Also wiederholen-fortlaufend-falls? oder -bis?*

A: *Du hast ja sowieso um alles herum eine wiederhole-fortlaufend-Schleife, damit du sozusagen es immer wieder aktualisieren kannst.*

Beispiel 3.3.4 *2. Studie, Gruppe AB:*

B: *Also beim Ultraschallsensor, wenn er darum fahren soll, dann muss man ihm sagen, er soll sich vor dem in Acht nehmen, was am nächsten dran ist. Denn es sind ja keine 2 m zwischen der Kiste und dem (Schrank). Ja, das gibt's bestimmt ja als Ding [Block in Programm].*

B: *Da muss man ihm sagen, dass [grübelt] und dann müssen wir ihm sagen, dass er sich auch nicht mehr als 10 cm davon entfernen sollte. Das er halt darum fährt.*

A: *Aber woher weiß er dann, wann er um die Ecke fahren muss?*

B: *Na ja, was ich eben meinte mit den 10 cm Abstand vielleicht [überlegt]. Also das, was am nächsten an ihm dran ist, soll nicht 10-15 cm verlassen.*

Beispiel 3.3.5 *2. Studie, Gruppe DE:*

E: *Er fährt auf die Kiste zu.*

D: *Nein, wir stellen ihn einfach neben die Kiste, dann dreht er sich bis er die Kiste findet. Wenn er die Kiste findet, dann dreht er sich um 90 Grad. Und dann fährt er ...*

E: *Lass uns machen er fährt, dann prüft er wieder, ob die Kiste noch da ist, wenn die Kiste da ist, dreht er sich [zeigt nach links] und fährt weiter. Wenn die Kiste nicht mehr da ist, dann fährt er in die Richtung... hm. Naja, er dreht sich ja um 90 Grad, dann fährt er kurz in diese Richtung weiter und dreht sich und wenn er die Kiste sieht (dreht er sich um) 90 Grad. Also er schaut, Kiste da, fahren [nach rechts gezeigt]. Kiste nicht da, fahren [zeigt gerade aus]. Kiste da [deutet auf Roboter guckt nach links], fahren [nach rechts drehen und gerade aus fahren].*

D: *Hä, wie Kiste nicht da? Der dreht sich doch komplett.*

E: *Er fängt so an [frontal vor Kiste gedeutet]. Er sieht die Kiste.*

D: *Warte. Das ist jetzt die Kiste. [legt sich Blatt hin zum verdeutlichen des Aufbaus]*

E: *Der kommt an und sieht die Kiste. Hält nach einem Abstand an. Dann dreht er sich 90 Grad [nach rechts], fährt kurz weiter, dreht sich [nach links], Kiste ist da (vor), fährt weiter [gerade aus], dreht sich [links], Kiste nicht mehr da [an Ecke vorbei], fährt vorwärts, dann dreht er sich wieder [links], Kiste ist da (vor), dann kann er das weitermachen [den Algorithmus]. Wenn die Kiste nicht da ist, dreht er sich immer um 90 Grad.*

D: *Ah ja! Okay also brauchen wir erstmal eine Schleife.*

Probleme antizipieren und teilen

Definition: Umfasst die Beschreibung von möglichen Problemen noch bevor sie auftreten. Das Teilen von Problemen und Suchen nach Lösungen kann z. B. in Online-Foren vorgenommen werden, was in diesen Studien nicht gestattet war.

Beispiel 3.3.6 *4. Studie, Gruppe CD:*

C: *Bei dem Widerstand gibt es keine Vorgaben wie der einzubauen ist? Wie bei der LED?*

Beispiel 3.3.7 *3. Studie, Gruppe CD:*

C: *Na er muss so lange gerade aus fahren bis ein Hindernis kommt, dann nach rechts, dann gerade aus.*

D: *Na gucke mal, dann fährt der hier rechts [um die Ecke, zeigt darauf] und dann sieht er ja kein Hindernis und fährt die ganze Zeit lang gerade aus [weg von der Kiste].*

C: *Ach so.*

D: *Das ist das Problem. Dann machen wir mit dem Maximum des Sensors, dass wenn in 2 m Entfernung nichts kommt, dann soll er nach (links?) gehen oder sowas. Verstehst du?*

Ziel in Teilaufgaben gliedern

Definition: Beschreibt das Zerteilen der Aufgabenstellung in einzelne Etappen, die durch ihre Abarbeitung zum Zielzustand führen.

Beispiel 3.3.8 *2. Studie, Gruppe DE:*

E: *Wir machen das jetzt mal komplett ohne Schleifen.*

D: *Wie ohne Schleife?*

E: *Na wir müssen erstmal (gucken) wie der es erkennt. Wir machen einfach der bleibt dann stehen. So ganz grundlegend, ob es mit dem Sensor überhaupt funktioniert.*

D: *Also erstmal fährt er?*

E: *Ja mach fahren.*

Auswahl von Input, Output und Verarbeitungsgeräten

Definition: Umfasst die Auswahl von Sensoren, Aktuatoren und Mikrocontrollern oder Mini-Computern entsprechend der Zielsetzung. Dazu zählt weitere Peripherie, die für die Kommunikation und Konstruktion zuvor genannter Komponenten benötigt wird (z. B. Steckbretter oder Klebstoff).

Beispiel 3.3.9 *4. Studie, Gruppe EF:*

E: *[...] erst mal die Lampen [...]. Rot, grün und gelb, oder?*

F: *Ja.*

E: *Müssen da Widerstände mit ran?*

F: *10 kΩ glaube ich. [...]*

E: *Der ist für den Fotosensor. Nicht für die Lampen.*

Programmieren/Konstruieren

Definition: Beschreibt das aktive Programmieren des Geräts bzw. den Auf- und Umbau von Geräten sowie Komponenten für Input und Output an den Geräten. Sofern parallel programmiert wird, sind auch die Kommentare in dieser Teilphase enthalten, die das Vorgehen planen.

Beispiel 3.3.10 *2. Studie, Gruppe DE: [programmieren]*

D: *Und wenn er etwas findet, bleibt er stehen.*

E: *Da müssen wir aber wieder so eine wenn-Funktion machen.*

D: *Also doch eine Schleife. [überlegen, was in die Schleife muss]*

D: *Nee wir brauchen einen Schalter.*

E: *Jetzt packst du das Gelbe [Steuerblock] da einfach rein.*

D: *Warte, wir müssen das erst wieder umstellen.*

D: *Okay, wenn er etwas erkennt...*

E: *Dann soll erstehen bleiben. Mach bei Häkchen 0 rein.*

Beispiel 3.3.11 *4. Studie, Gruppe IJ:*

J: *Lampen sind jetzt alle soweit fertig? Ach so an den Digitalpin anschließen.*

I: *Ja.*

J: *Welche haben wir denn? Einfach 13, 12 und 11.*

I: *Ok.*

J: *Nimm mal längere Kabel.*

I: *Hast du schon schwarz?*

J: *Können auch unterschiedliche Farben nehmen.*

I: *Dann können wir es besser auseinanderhalten. Hier.*

J: *Das ist 13, das ist... Ja. Gibt es noch ein drittes langes Kabel?*

I: *Wollen wir nicht das hier nehmen?*

J: *Das ist nicht so toll.*

I: *Und das Gelbe...*

J: *Das ist länger. Nimm das. [bauen beide]*

Programm ausführen

Definition: Beschreibt den Start des Programms (bzw. eines Teilprogramms) oder Experiments durch die SuS.
Anmerkung: Codiert wurden vor allem Handlungen der SuS, die durch die transkribierende Person beschrieben wurden (ohne verbale Äußerung durch die SuS).

Beispiel 3.3.12 *3. Studie, Gruppe AB:*

[*Probieren aus; stellen Roboter direkt weiter nach vorne Richtung Kiste; fährt die Längsseite* [...] *ab, fährt an der ersten Ecke jedoch wieder zurück und im Kreis - von Kiste zu weit weg*]
[*Brechen ab*]

Beispiel 3.3.13 *4. Studie, Gruppe CD:*

C: *Gut. Searching board. Dafür scheint er wieder mal eine Weile zu brauchen. Das kennen wir ja. Was machen wir, wenn er den Arduino nicht findet? Wir versuchen ihn mal zu reseten. Wunderbar. Hat funktioniert. So wir hatten bei analog 0 jetzt einen Widerstand.* [...]

Beobachtung der Auswirkungen

Definition: Umfasst das aktive Beobachten der Programmausführung durch die SuS bis das Programm beendet ist. Das Abbrechen des Programms während der Ausführung kann ebenfalls ein Bestandteil sein.
Anmerkung: Dieser Teil wurde ebenfalls nicht zwangsläufig von den SuS verbalisiert.

Beispiel 3.3.14 *4. Studie, Gruppe EF:*

E: *Also eigentlich müsste es... ändert sich der Wert, wenn ich den Finger raufhalte?*

F: *Ganz langsam. Er ist fast still.*

E: *Ja. Das soll ja nicht passieren. Ah er geht höher.*

E: *500. 200. 500. 200.*

F: *Ja.*

Vergleich der Auswirkungen mit Programmteilen

Definition: Beschreibt die Reflexion der SuS über die Aktionen des Geräts und durch welche Teile ihres geschriebenen Programms diese verursacht wurden.

Beispiel 3.3.15 *3. Studie, Gruppe AB:*

[*Überlegen zusammen bei den Problemursachen*]

B: *Ultraschallsensor kaputt? Nee, er hat es teilweise einfach nicht wahrgenommen oder spät, obwohl wir 40 cm hatten. Er stand ja teilweise 20 cm entfernt und hat es nicht wahrgenommen.* [...]

A: *Ich glaube schon, dass der Sensor funktioniert.* [*programmieren weiter*]

A: *Wo war jetzt das Problem?*

B: *Er hat sich teilweise sehr weit nach links gedreht. Guck mal da rein, also bei der ersten Linksdrehung.*

A: [*Gehen Programm durch*] *dann müssen wir es hier auch ein bisschen runter stellen (Linksdrehung) -40.*

Identifikation des Problems

Definition: Beschreibt die explizite Benennung eines Problems bzw. der Ursache des Problems.

Beispiel 3.3.16 *2. Studie, Gruppe C:*

[*Probiert erneut aus*]
[*Bricht Programm ab*]

C: *Da ist das Sichtfeld (des Ultraschallsensors) stark eingeengt, also nicht 90 Grad.*

Beispiel 3.3.17 *4. Studie, Gruppe GH:*

H: *Jetzt passiert was.*

G: *Ja. Sehr schön. [LED leuchtet]*

G: *Was hast du anders gemacht?*

H: *Ich habe es [einen Pin] einfach in die Mitte gesteckt. Oh. Ist das viel [hoher Wert]*

G: *Ok.*

Entscheidung eine Variable zu ändern

Definition: Beinhaltet die Entscheidung der SuS über eine Veränderung, die im nächsten Schritt vorgenommen werden sollte, um dem Zielzustand näher zu kommen.

Beispiel 3.3.18 *4. Studie, Gruppe GH:*

H: *Wenn ich mich jetzt hier wegbewege, dann ist es natürlich viel mehr. Es ist gerade bei 65. Aber ist ja egal. Solange es erst mal funktioniert.*

G: *Wir können ja 70 einfach machen.*

Beispiel 3.3.19 *4. Studie, Gruppe IJ:*

J: *Mal gucken, ob es besser wird. [J leuchtet mit Lampe; I guckt auf Zettel]*

J: *Habe ich mich vertan?*

I: *Ähm.*

J: *Muss ich die Zeichen einfach nur ändern? Kann das sein? Das ist jetzt größer als 400. Genau es muss kleiner als 400.*

I: *Oh.*

Entscheidung für erneuten Start der (Teil-)Schleife

Definition: Umfasst die Wiederholung von Phasen oder Teilphasen, nachdem die Hauptphasen bereits einmal durchlaufen wurden.

Beispiel 3.3.20 *3. Studie, Gruppe EF:*

[*Probieren aus; F startet Programm mit Hand vor dem Sensor, Roboter dreht sich direkt nach links, fährt kurz gerade aus, nach rechts, kurz gerade aus, nochmal nach rechts und bleibt vor der Kiste stehen*]
[...]
[*Programm ist zu Ende*]

E: *Was?*

F: *Hä...*

E: *Hast du auch das richtige Programm eingestellt?*

F: *Ja.*

[*Startet Programme erneut, fährt nun kurz geradeaus weiter, als die Hand von F schon weg ist, dreht nach rechts, kurz geradeaus weiter, wieder nach rechts und bleibt stehen*]

Das zuvor vorgestellte theoretische Modell konnte somit durch empirische Daten gestützt und zugleich angepasst werden. Daraus ergeben sich leichte Verschiebungen im Modell (siehe Abbildung 3.18). Eine Veränderung beinhaltet, dass das *Gliedern eines Problems in Teilaufgaben* in der Praxis eher in der Evaluationsphase als in der Vorbereitungsphase erfolgt. Das ist insbesondere für den ersten Durchlauf dieses Kreislaufs wichtig, da die SuS nicht mit ausreichend Planung vorgehen, um das Problem direkt zu zerteilen und die Antizipation von Problemen noch nicht sehr ausgeprägt zu sein scheint. Zumeist fällt erst zu einem späteren Zeitpunkt (nach mehreren Durchläufen des Zyklus) die Entscheidung, das Problem in Teilaufgaben zu teilen. Dies wurde in der Evaluationsphase oftmals als logischer Schluss daraus beobachtet, dass ihr Programm zu unübersichtlich und umfangreich wurde, um einen spezifischen Fehler zu finden. Zusätzlich wurde eine weitere Adaption vorgenommen, indem ein bisher nicht berücksichtigter Aspekt hinzugefügt wurde. Nicht eindeutig einer Teilphase zugeordnet, aber bei der Durchführung und Evaluation angesiedelt, befindet sich der Umgang mit *Unsicherheiten und die Identifikation von Fehlern.* Hierbei handelt es sich um die physikalische Komponente des Physical Computing, die bei der Beschreibung von Teilphasen in der Physical-Computing-Literatur bislang eine geringe Rolle spielt. Neben Problemen in der Programmierung und Konstruktion, die recht geläufig in der Informatik sind, müssen beim Physical Computing nun auch bekannte Probleme aus der Physik berücksichtigt werden. Durch die Arbeit mit Sensoren ist die Thematisierung und der Umgang mit Messunsicherheiten und Fehlern hinzugekommen. Die Beschreibung dieser Teilphase und ein zugehöriges Beispiel dafür wird im Folgenden dargestellt.

Unsicherheiten und Fehler identifizieren

Definition: Beschreibt die Evaluation der SuS von aufgetretenen Messunsicherheiten oder Fehlern, die technischer oder physikalischer Natur sind. Die gewollte oder versehentliche Intervention durch den Menschen ist ein weiterer Faktor, der zu Fehlern und Messunsicherheiten führen kann.

Beispiel 3.3.21 *3. Studie, Gruppe CD:*

[*Starten Programm*]
[*Roboter fährt gerade auf die Kiste zu und dagegen, fährt sich fest*]

C: *Nee, funktioniert nicht.* [*brechen Programm ab*] [...]

C: *Also, was sind unsere Beobachtungen?* [*C füllt Hilfsblatt aus*]

C: *Falsch programmiert [als Antwort].*

D: *Ultraschallsensor könnte kaputt sein?*

C: *Könnten es auch die Lichtverhältnisse sein?*

D: *Nee, ist ja kein …*

C: *Und nicht im Blickfeld vielleicht? Nee!*

Die SuS nahmen die Messunsicherheiten und Fehler selten bewusst wahr, jedoch konnten die Probleme an der benannten Stelle innerhalb des Physical-Computing-Prozesses beobachtet werden. Dabei handelt es sich z. B. um gewisse Unsicherheiten in der Messgenauigkeit bei der Arbeit mit Sensoren. Dies kann dazu führen, dass das gleiche Programm mit ähnlichen Rahmenbedingungen in der Ausführung nicht zum gleichen Ergebnis kommt.

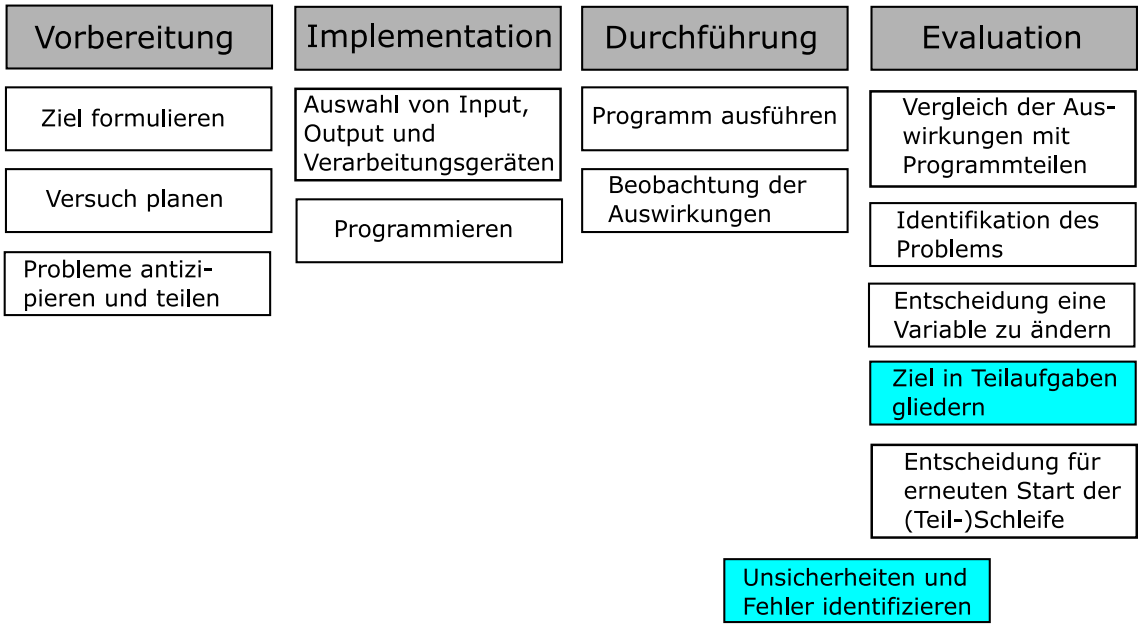


Abbildung 3.18: Angepasste Physical-Computing-Modell nach Schulz und Pinkwart (2016)

Für dieses Modell gilt, ähnlich wie bei der wissenschaftlichen Erkenntnisgewinnung (vgl. Modell von Schreiber et al., 2009), dass die Phasen und Teilphasen nicht in einer festen Reihenfolge ablaufen müssen bzw. komplett übersprungen werden können. Es kann geschlussfolgert werden, dass der Physical-Computing-Prozess und der Prozess der wissenschaftlichen Erkenntnisgewinnung theoretisch und empirisch parallel verlaufen. Bislang sind diese Ergebnisse jedoch noch auf die verwendeten Gerätetypen und Aufgaben beschränkt, wobei im Rahmen dieser Arbeit bereits zwei sehr verbreitete Gerätetypen untersucht wurden. Im Vergleich zu dem theoriegeleiteten Modell des Physical Computing (vgl. Tabelle 3.1) fällt auf, dass die Gliederung des Prozesses in Teilaufgaben nicht bereits in der Vorbereitungsphase geplant wird, sondern zumeist in der Evaluationsphase umgesetzt wird. In vielen Fällen entstand diese Entscheidung daraus, dass die SuS die Aufgabe

als zu schwer einschätzten, um sie als Ganzes zu lösen. Es erscheint jedoch sinnvoll, die SuS anzuleiten, ihren Lösungsweg schon vor der Evaluationsphase in Teilziele zu gliedern, um Frustration zu vermeiden. Eine Teilphase wurde auf Grundlage der empirischen Untersuchung zu dem Physical-Computing-Prozessmodell hinzugefügt. Diese adressiert die Identifikation und den Umgang mit Unsicherheiten und Fehlern. Dabei handelt es sich vor allem um Unsicherheiten, die ihren Ursprung in der Aufnahme und Verarbeitung von Sensordaten haben. Da der Umgang mit Daten ein wichtiger Bestandteil der Physical-Computing-Definition ist, kann davon ausgegangen werden, dass die ermittelte Teilphase im starken Zusammenhang mit dem Erfolg von Physical-Computing-Aktivitäten steht.

3.3.2 Untersuchung der Phasenübergänge im Physical-Computing-Prozess

Um den Physical-Computing-Prozess detailliert beschreiben zu können, sind jedoch nicht nur die Bestandteile des Prozesses von Bedeutung, sondern auch der zeitliche Verlauf dieser Komponenten. Dadurch kann ähnlich der Code-Relation-Matrix (vgl. Abbildung 3.11) festgestellt werden, welche Phasen in Abhängigkeit zueinander stehen. Darüber hinaus kann ermittelt werden, was mögliche Gründe sein können, aus denen von dem idealtypischen, zyklischen Prozess abgewichen wird und sich z. B. kleine Kreisläufe innerhalb des Gesamtprozesses ergeben. Für die Analyse des Phasenverlaufs im Physical Computing werden sogenannte *Codelines* verwendet, die durch ein Werkzeug von MaxQDA für ausgewählte Codes erzeugt werden können. Die Codelines werden genutzt, um die Übergänge von einer Phase in eine andere zu modellieren, wobei anschließend auf die Transkripte zurückgegriffen und der Kontext des Phasenwechsels betrachtet wird.
Bei dieser Auswertung werden die Paragraphen eines Transkriptes in zehn Abschnitte geteilt (x-Achse) und deren auftretende Codes betrachtet (y-Achse). Abhängig von der individuell variierenden Interventionsdauer und der Geschwindigkeit der ProbandInnen umfassen die Transkripte eine unterschiedliche Anzahl von Paragraphen. Die entstehenden Codelines werden von links nach rechts gelesen und beschreiben dabei den Phasenwechsel über die gesamte Bearbeitungszeit für jeweils eine Gruppe. In einigen Gruppen endet die Codeline, obwohl die Paragraphen zeigen, dass das Video weiterlief (vgl. Abbildung 3.24 und 3.25). In diesen Fällen wurde ein ausführliches Abschlussgespräch mit den Gruppen geführt, welches jedoch nicht codiert wurde, da die Bearbeitung von Physical-Computing-Aufgaben zu diesem Zeitpunkt abgeschlossen war.
Da die Studie in Abbildung 3.20 mit einer ungeraden Anzahl von SuS stattfand, beschreibt sie den Phasenverlauf einer einzelnen Person. In diesem Verlauf zeigen sich keine deutlichen Unterschiede zu dem von den paarweise arbeitenden Gruppen (vgl. Abbildungen 3.21 oder 3.22).
Zunächst werden die Codelines der 2. Studie betrachtet (Abbildung 3.19, 3.20, 3.21), in der die SuS mit LEGO Mindstorms-Robotern arbeiteten. Die zuvor beschriebenen

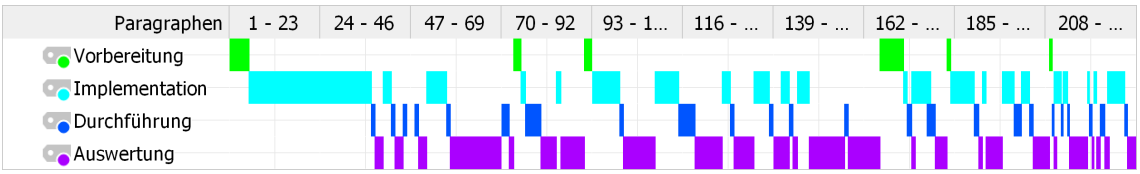


Abbildung 3.19: Codeline zur Beschreibung der Hauptphasen, Studie 2, Gruppe AB

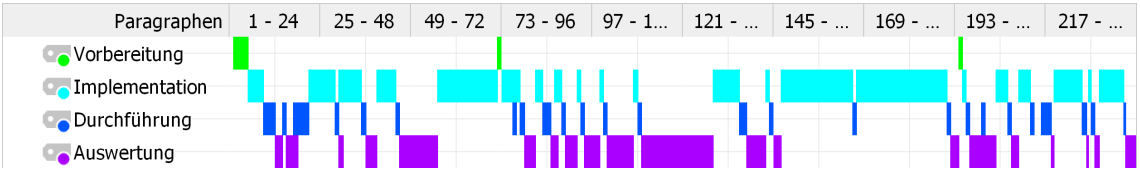


Abbildung 3.20: Codeline zur Beschreibung der Hauptphasen, Studie 2, Gruppe C

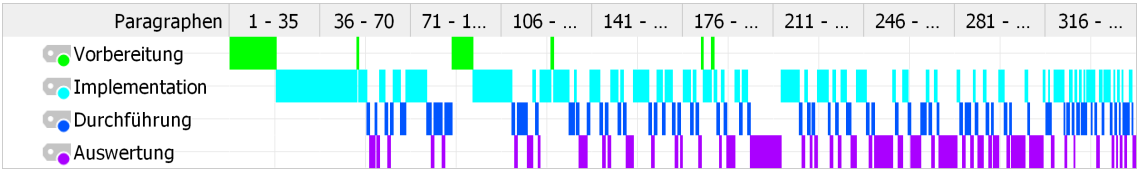


Abbildung 3.21: Codeline zur Beschreibung der Hauptphasen, Studie 2, Gruppe DE

Phasen des Physical-Computing-Prozesses konnten bei den Gruppen beobachtet werden. Alle Gruppen starteten mit einer Vorbereitungsphase, in der die Aufgabenstellung geklärt und das Vorgehen (teilweise nur in Ansätzen) geplant wurde. Es schließen sich bei allen Gruppen zunächst die Implementations- und die Durchführungsphase an. In diesem ersten Durchlauf des Gesamtprozesses wurde die Durchführung direkt evaluiert. Nach weiteren Durchführungsphasen ist auffällig, dass die anschließende Auswertung teilweise übersprungen wird.

Die Vorbereitungsphase findet deutlich seltener statt als die anderen Phasen. Sie dient jedoch nicht nur am Anfang zur Klärung der Aufgabenstellung, sondern es wird sich auch während fortgeschrittener Aufgabenbearbeitung erneut auf die Aufgabenstellung bezogen. Diese wird gegebenenfalls geklärt oder es wird ein gänzlich neuer Lösungsansatz gefunden. In allen Studien bekamen die SuS hauptsächlich eine Aufgabe und maximal eine zweite Aufgabe. Da in allen Codelines der 2. Studie mehr als zwei Vorbereitungsphasen auftraten, wird die These gestützt, dass sich alle Phasen zyklisch wiederholen.

Die Codelines zeigen, dass die Gruppen AB und DE zunächst eine ausgedehnte Implementationsphase durchliefen, in der das Programm aufgebaut wurde. Die weiteren Implementationsphasen sind eher kurz, was darauf hindeutet, dass nur noch kleine Veränderungen vorgenommen wurden, z. B. die Änderung der Werte von Variablen. Möglicherweise haben sie auch nur den Quellcode nachvollzogen und damit theoretisch überprüft, ob ihr Programm funktionieren kann. Gruppe C durchlief diese ausgedehnte Phase nicht innerhalb der ersten Implementationsphase, sondern erst zu einem späteren

Zeitpunkt. Des Weiteren fanden bei Gruppe C mehrere ausgedehnte Implementationsphasen statt, was damit in Zusammenhang zu bringen ist, dass die Aufgabe zunächst falsch verstanden wurde und dann größere Änderungen am Programm vorgenommen werden mussten. In der Gruppe AB ist zu erkennen, dass einige Implementationsphasen direkt ohne eine Erprobung zu einer Auswertungsphase führten. In diesem Fall bezogen sich die SuS auf vorherige Beobachtungen oder Evaluationen und antizipierten mögliche Probleme bzw. setzten ihre Beobachtungen in einen neuen Kontext und vervollständigten frühere Auswertungsansätze. Aus diesem Grund war eine Erprobung nicht notwendig und wurde ausgelassen. Gruppe C und EF gingen an einigen Stellen nach der Implementation zurück in die Vorbereitungsphase und dann erneut in die Implementation. Dieses Verhalten deutet darauf hin, dass die Fragestellung bzw. das Problem detaillierter geklärt werden musste, weil sich Unklarheiten während der Implementation zeigten. Es ist auffällig, dass die Implementationsphase teilweise deutlich länger ist als andere Phasen. Das kann einerseits damit zusammen hängen, dass diese Phase durch die Eingaben am Computer per se zeitintensiver ist und mit den Vorerfahrungen der SuS beim Programmieren. Andererseits ist sie ebenfalls von Planungsinhalten gekennzeichnet, ohne dass zwangsläufig eine gemeinsame Diskussion stattfand oder die Diskussion nicht von der Implementation getrennt betrachtet werden konnte.

Die Phase der Durchführung grenzt üblicherweise an eine vorherige Implementation und eine anschließende Auswertung an. Innerhalb der Durchführungsphase kann es jedoch zu mehreren Erprobungen gekommen sein, ohne dass Änderungen am Programmcode oder der Umwelt vorgenommen wurden. Das kam üblicherweise vor, wenn die SuS die Ausführung des Programms erneut sehen wollten, um Schlüsse ziehen zu können bzw. wenn sie sich unsicher waren, wie sie fortfahren sollten.

Die Evaluationsphase schließt zumeist an eine Durchführungsphase an und ist inhaltlich sehr unterschiedlich ausgeprägt, was auch in der Phasenlänge zu erkennen ist. Insbesondere in kurzen Phasen war es kaum möglich, dass die SuS einen Ursache-Wirkungs-Zusammenhang erläuterten, sondern nur eine Bewertung äußerten. Das wird durch Formulierungen wie „nein“ oder „das hat nicht geklappt“ belegt. Die Phasenverläufe zeigen auf, dass es üblich ist, dass die SuS nach einer Evaluationsphase direkt in die nächste Implementationsphase übergehen. Dabei wird selten die allgemeine Problemstellung behandelt und eine weitere Vorbereitungsphase durchlaufen. Es handelt sich meist um Teilprobleme und damit um Subprozesse, bei denen z. B. die Evaluation beschreibt, welche Drehungen und welche Variablen verändert werden müssen. Somit ist im Anschluss nur die Implementation der Werte nötig, ohne das Vorgehen explizit zu planen. Teilweise (z. B. bei Gruppe C) ist ein Übergang der Evaluationsphase in die Durchführungsphase zu beobachten. Das war der Fall, wenn den SuS bewusst wurde, dass sie die Durchführung erneut beobachten müssten, um Schlüsse ziehen zu können.

Bei der Gruppe DE zeigt sich ein klar strukturiertes und zyklisches Durchlaufen des Gesamtprozesses. Da ab dem zweiten Drittel alle Phasen circa die gleiche Länge und

nahezu die gleiche Abfolge aufweisen, ist davon auszugehen, dass die SuS viele kleine Änderungen vornahmen und diese zügig ausprobierten.

Aus der 4. Studie mit Arduino-Mikrocontrollern sind die folgenden Phasenverläufe codiert worden (Abbildungen 3.22, 3.23, 3.24, 3.25, 3.26). Es handelt sich dabei um fünf Gruppen, die ebenfalls eine bis maximal zwei Aufgaben lösten. Diese Verläufe werden nun mit der Analyse der LEGO Mindstorms-Roboter verglichen, um die Überschneidungen der beiden Prozesse auf unterschiedliche Gerätetypen zu stützen. Des Weiteren werden die Unterschiede innerhalb dieser Gerätevielfalt auch in den folgenden Forschungsfragen Berücksichtigung finden.

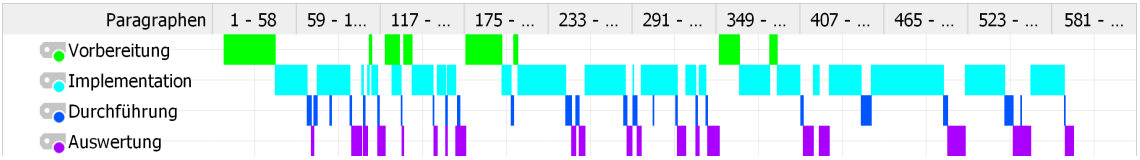


Abbildung 3.22: Codeline zur Beschreibung der Hauptphasen, Studie 4, Gruppe AB

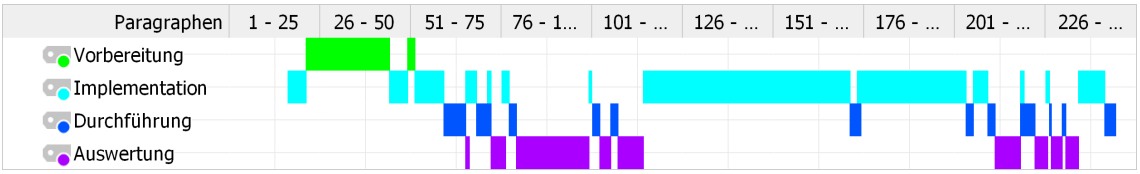


Abbildung 3.23: Codeline zur Beschreibung der Hauptphasen, Studie 4, Gruppe CD

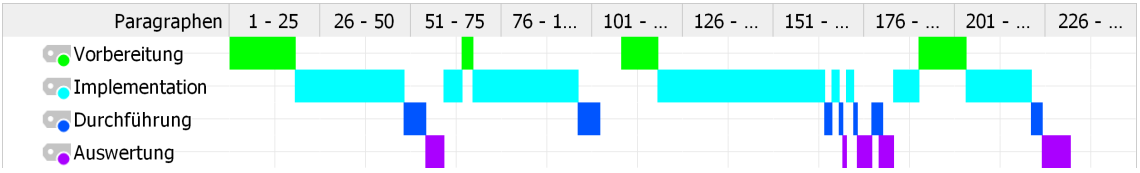


Abbildung 3.24: Codeline zur Beschreibung der Hauptphasen, Studie 4, Gruppe EF

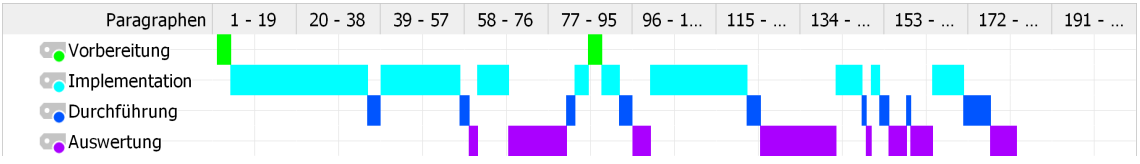


Abbildung 3.25: Codeline zur Beschreibung der Hauptphasen, Studie 4, Gruppe GH

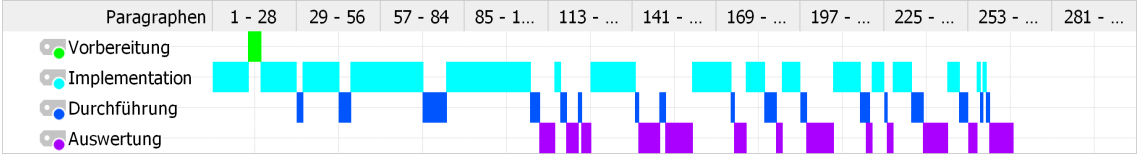


Abbildung 3.26: Codeline zur Beschreibung der Hauptphasen, Studie 4, Gruppe IJ

Während der Arbeit mit Arduino-Mikrocontrollern durchlaufen die SuS die gleichen vier Hauptphasen wie die SuS bei der Arbeit mit LEGO Mindstorms-Robotern. Die Phasenübergänge zeigen das gleiche Muster bzw. die gleichen Möglichkeiten des Übergangs in eine folgende Phase. Es ist auffällig, dass in der Studie mit den Arduino-Mikrocontrollern die Phasen der Implementation einen großen Anteil einnehmen und diese Phasen teilweise sehr lang sind (vgl. Abbildungen 3.22, 3.24 und 3.25). Dieser Code-Verlauf ist dadurch zu erklären, dass die Veränderungen im Programm (in der Implementationsphase) ohne expliziten Phasenwechsel durch die SuS unmittelbar auf dem Mikrocontroller ausgeführt werden. Aus diesem Grund war es bei der Codierung dieser Phasen nicht immer eindeutig zuzuordnen, ob die SuS gezielt in die Durchführungsphase wechselten.
Bei den Gruppen S4-AB und S4-EF, dargestellt in den Abbildungen 3.22 und 3.24, sind in der zweiten Hälfte des Prozesses längere Implementationsphasen zu beobachten. Das hängt damit zusammen, dass beide Gruppen eine 2. Aufgabe bewältigten und Gruppe AB dies nicht explizit plante. Hingegen durchliefen S4-EF direkt zum Beginn der 2. Aufgabe eine Vorbereitungsphase.
Bei der Betrachtung aller Gruppen wurden weitere Auffälligkeiten der Arbeitsabläufe entdeckt. Wenige Gruppen starteten nicht mit einer Vorbereitungsphase, sondern stiegen direkt mit der Implementation ein. War das der Fall, dann wurde die Vorbereitungsphase direkt nach maximal einem Durchlauf des Physical-Computing-Prozesses in Form einer ausgedehnten Vorbereitungsphase nachgeholt (vgl. Abbildungen 3.26 und 3.27). Eine weitere Gruppe durchlief hingegen eine ausgedehnte Vorbereitungs- und Implementationsphase (vgl. Abbildung 3.28), wobei sie dafür im Anschluss kürzere Implementationsphasen durchlief als z. B. bei den Gruppen S2-C (Abbildung 3.20) und S4-AB (Abbildung 3.22) zu erkennen sind.

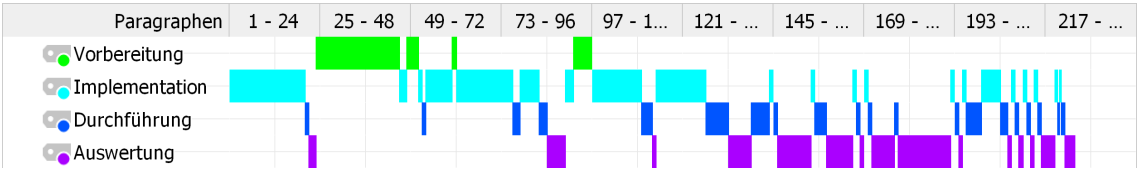


Abbildung 3.27: Codeline zur Beschreibung der Hauptphasen, Studie 1, Gruppe AB

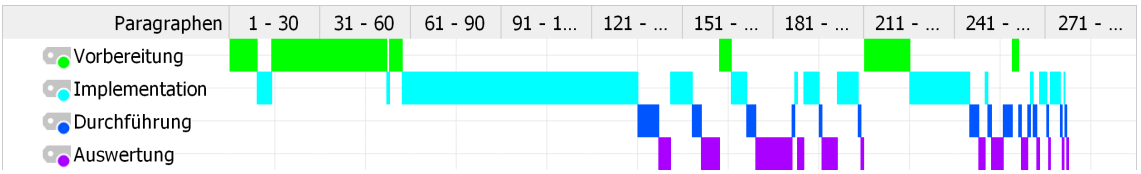


Abbildung 3.28: Codeline zur Beschreibung der Hauptphasen, Studie 1, Gruppe CD

Auf Gruppe S4-IJ trifft diese Beobachtung nicht zu (vgl. Abbildung 3.26). Diese führte lediglich eine kurze Vorbereitungsphase durch und begann erst zu einem vergleichsweise späten Zeitpunkt ihre Untersuchungen zu evaluieren.

Über alle Gruppen hinweg ist zu beobachten, dass nicht alle Durchführungsphasen evaluiert wurden bzw. die Evaluation zum Teil sehr kurz ausfiel.

Auf Grundlage der Daten der Codelines, wurde ein Modell entwickelt, das den Phasenverlauf des Physical-Computing-Prozesses beschreibt und die Übergänge von Phasen, ausgehend von der Evaluationsphase, skizziert. Es wurde das Augenmerk auf die Evaluationsphase gelegt, da in den Codelines bereits ersichtlich ist, dass die Evaluationsphase übersprungen wird oder kurz ausfällt. Somit werden die Phasenübergänge nach der Evaluation zuerst untersucht, um zu ermitteln, wie ein iterativer Physical-Computing-Prozess ablaufen kann, der alle Hauptphasen umfasst. Es gibt drei mögliche Wege, wie nach der

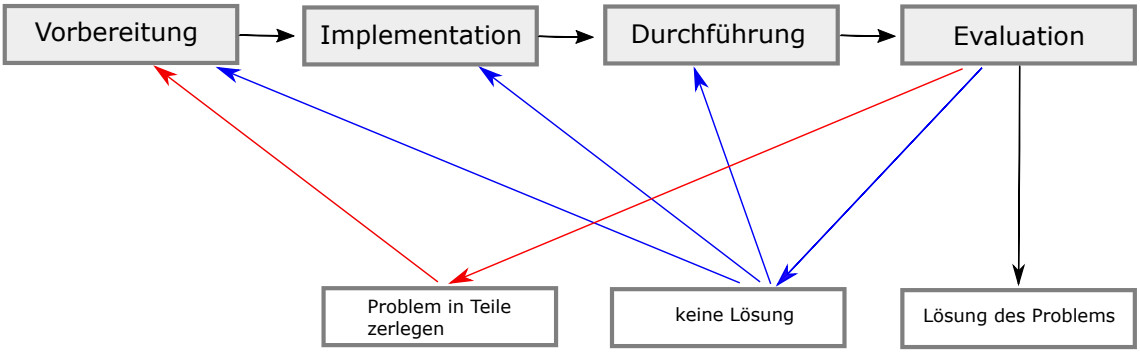


Abbildung 3.29: Phasenverlauf nach der Evaluationsphase im Physical-Computing-Prozess von Schulz und Pinkwart (2016)

Evaluationsphase weiter vorgegangen werden kann. Der erste und eher theoretische Fall ist, dass nach einem Durchlauf des Physical-Computing-Prozesses die Lösung des Problems bzw. die Aufgabe gelöst wurde (schwarzer Pfeil, Abbildung 3.29). Dies ist jedoch unwahrscheinlich und würde bedeuten, dass die SuS von Beginn an das Objekt richtig konstruiert und programmiert haben. Der zweite Fall ist, dass in der ersten Evaluationsphase noch keine Lösung gefunden wurde. Er beschreibt den Übergang in die Phase der Durchführung, Implementation oder Vorbereitung (blaue Pfeile, Abbildung 3.29). Der Schritt in die Durchführungsphase wurde zumeist gewählt, wenn die SuS das Programm auf dem Roboter/Mikrocontroller erneut starteten, um die Auswirkungen ein weiteres Mal beobachten zu können und anschließend zu evaluieren.

Beispiel 3.3.22 *4. Studie, Gruppe AB:*

B: *Warum bewegt der sich noch?*

A: *Du musst ihn erst mal ausschalten.*

B: *Ja. Das ist vielleicht eine gute Idee.* [*schaltet aus und das Neue an*]

B: *Ok. Und jetzt.*

A: *Dreht er sich nicht mehr.*

B: *Ach so. Ja. Weil er geht erst durch die [zeigt auf eine Schleife am Bildschirm] Schleife durch. [testet noch mal]*

B: *Warum dreht der sich nicht? [betrachten die Konstruktion und überlegen]*

B: *Warte mal kurz. Wie ist das orientiert? Also wo ist da clock-wise und wo counter-wise?*

A: *So und so. [zeigt mit dem Finger im und gegen den Uhrzeigersinn] Da hast du keine Orientierung direkt.*

B: *Ach so stimmt. Das ist anti-clockwise und das dementsprechend... also so ist anti-clockwise und so ist clockwise. [zeigt es mit dem Finger, überlegen]*

Das Beispiel 3.3.22 veranschaulicht, wie die SuS versuchten einen Servo-Motor mit einem Arduino-Mikrocontroller anzusteuern. Dabei starteten sie zuerst das Programm, wobei der Motor keine Aktion ausführte. Sie starteten das Programm mehrfach und analysierten dabei verschiedene Komponenten, die ein Problem darstellen könnten. Dabei nahmen sie keine Änderungen vor.
Von der Evaluationsphase wurde direkt in die Implementation übergegangen, um das Programm bzw. die Roboterkonstruktion zu überarbeiten, was auch *Trial-and-Error* implizieren kann. Der dritte Fall war der Übergang in die Vorbereitungsphase, in der dann zum Beispiel die Hypothese überarbeitet wurde. Die roten Pfeile (Abbildung 3.29) zeigen den Übergang an, wenn in der Evaluation die Entscheidung getroffen wird, das Problem in Teilprobleme bzw. -aufgaben zu gliedern. Dies führt dann zurück in die Vorbereitungsphase, da das Vorgehen geplant werden und neue Hypothesen bzw. Unterhypothesen gebildet werden müssen. In diesem Szenario läuft für jedes Subproblem der gezeigte Prozess ab. Dies geschieht so lange, bis das Teilproblem gelöst ist und das Nächste bearbeitet wird.

Beispiel 3.3.23 *2. Studie, DE:*

E: *Okay, hier hinten erkennt er was [Wand].*

D: *Okay, mache mal deine Hand davor. Näher, näher. Ja.*

E: *Also hier ist erstmal der Roboter [Kante vorne].*

D: *Ca. 10 [cm]. Nein. Mach mal deine Finger zusammen [vor dem Sensor]. Ja das ist so bei 18 cm. Also erkennen tut er was. Mach mal deine Hand ein bisschen höher, höher, höher... ja okay, das steigert sich nicht.*

E: *Wir machen das jetzt mal komplett ohne Schleifen.*

D: *Wie ohne Schleife?*

E: *Na wir müssen erstmal (gucken) wie der es erkennt. Wir machen einfach der bleibt dann stehen. So ganz grundlegend, ob es mit dem Sensor überhaupt funktioniert.*

Der Übergang von der Evaluationsphase in die Vorbereitungsphase kann an Beispiel 3.3.23 verdeutlicht werden. Die SuS bearbeiteten Aufgaben für einen LEGO Mindstorms-Roboter, bei denen sie den Roboter um eine Kiste fahren lassen sollten. Sie versuchten bereits mehrfach den Roboter um die Kiste fahren zu lassen, bis sie probierten einen geeigneten Schwellwert für den Ultraschallsensor zu ermitteln, d. h. ab wann der Roboter nah genug an der Kiste ist und sich drehen sollte. Daraufhin schlug E vor, die Funktionsweise des Ultraschallsensors grundlegend zu untersuchen und ohne weitere Einflussfaktoren wie z. B. Schleifen im Programm zu implementieren. Dadurch wird das Problem in Teile aufgegliedert, die einfacher zu bewältigen sind.
Darüber hinaus kann ein umfangreicheres Modell (in Abbildung 3.30) aus den Codelines abgeleitet werden, welches die Phasenübergänge aller Phasen berücksichtigt.

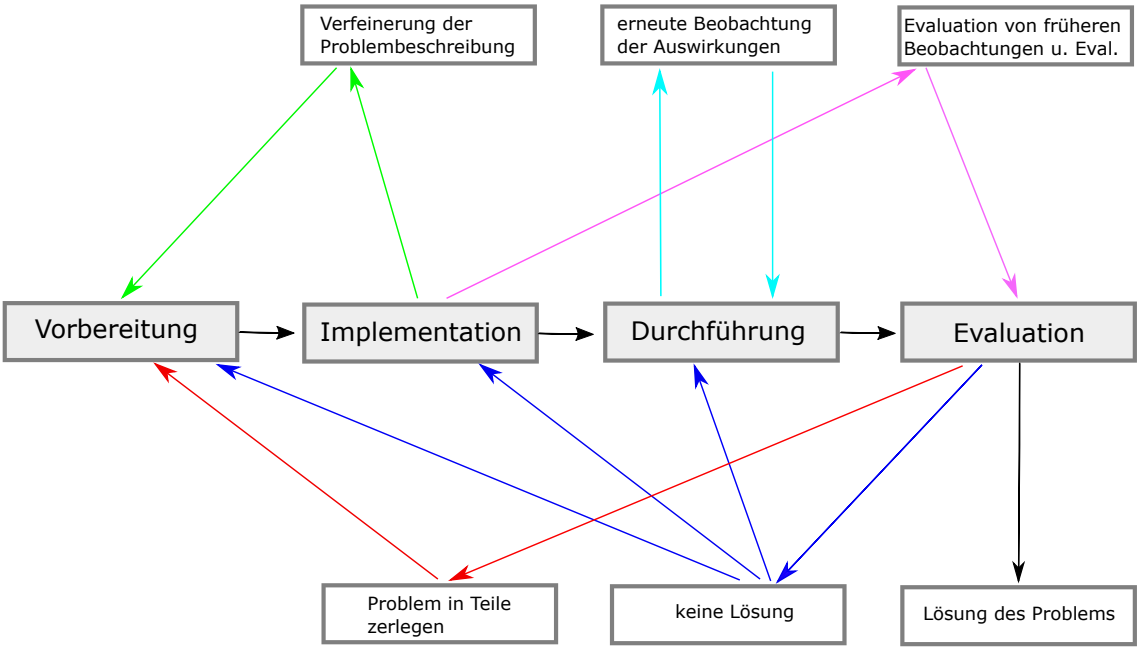


Abbildung 3.30: Gesamter Phasenverlauf im Physical-Computing-Prozess

Ausgehend von der Implementationsphase konnten zwei weitere mögliche Übergänge neben dem Idealtypischen zur Durchführungsphase festgestellt werden. Eine Möglichkeit besteht darin, nach der Implementationsphase in die Vorbereitungsphase zurückzukehren (vgl. grüne Pfeile, Abbildung 3.30). Wurde dieser Weg gewählt, nutzten die SuS die folgende Vorbereitungsphase meist, um das Problem detaillierter zu beschreiben. Vermutlich wurde ihnen bei dem Versuch der Implementation bewusst, dass die Klärung des Ziels unzureichend war, um einen Algorithmus daraus zu erarbeiten.

Beispiel 3.3.24 *3. Studie, AB:*

A: *Also er fährt jetzt irgendwie immer weiter von der Kiste weg.*

B: *Vielleicht war es doch ein bisschen zu viel drehen.* [*ändern wieder gleich im Programm*]

A: *Also erstmal das hier so ein bisschen runter. Vielleicht 84* [*Gradzahl der Drehung*]*?*

B: *Ja 54 - 84 so.*

A: *Und wenn wir jetzt weniger zurück drehen, weil er weiß ja, da muss irgendwo nach links. Weniger können wir ja ruhig, damit er sich nicht entfernt. War noch irgendwas?*

A: *Na 40 war eigentlich richtig, er ist nur weggefahren.* [...]

B: *Müsste es nicht der rechte sein (Motor), denn er dreht sich ja links rum.*

A: *Nee, wenn die Drehung nach außen stärker ist, dann ist der innen ja stärker.* [*gehen das am Roboter zusammen durch*] *Und diese Bewegung [innen] ist schwächer als diese [außen- rechts].*

In dem Beispiel 3.3.24 versuchten die SuS die Gradzahl einer Umdrehung zu adaptieren, nachdem ein Versuch mit einem LEGO-Roboter um eine Kiste zu fahren nicht erfolgreich war. Sie werteten den Versuch aus und stellen fest, dass sich der Roboter durch die Gradzahl der Drehungen von der Kiste zunehmend entfernte (Evaluationsphase). Im Anschluss implementierten sie direkt eine veränderte Gradzahl, die das festgestellte Problem beheben sollte. Dann gingen sie in die Evaluationsphase zurück und diskutierten, welcher Motor sich schneller drehen müsse, um eine Drehung nach links zu beeinflussen.
Als zweite Möglichkeit wird die Durchführungsphase übersprungen und die SuS gingen direkt in die Evaluationsphase über (pinke Pfeile, Abbildung 3.30). In diesem Fall evaluierten sie Durchführungsphasen aus vorherigen Prozessdurchläufen. Das war z. B. der Fall, wenn sie ihre aktuelle Vorgehensweise hinterfragten und bemerkten, dass sie das Gleiche oder Ähnliches zuvor bereits versucht hatten. Teilweise schienen die SuS vorherige Untersuchungen an dieser Stelle in einem neuen Kontext zu betrachten und evaluierten aufgrund dessen ihr Vorhaben. In den Übergängen der Durchführungsphasen wurde deutlich, dass mehrere Durchführungsphasen hintereinander stattfinden können, ohne dass die Beobachtungen oder Ähnliches zwischendurch explizit evaluiert wurden (türkise Pfeile, Abbildung 3.30). In diesem Fall führten die SuS das gleiche Programm erneut aus, ohne Änderungen vorzunehmen. Dabei konnten sie die Auswirkungen des Programms zum wiederholten Mal beobachten und es wurden ggf. Auswirkungen sichtbar, die zuvor nicht zu erkennen waren. Darüber hinaus kam es ebenfalls vor, dass die SuS bewusst eine wiederholte Ausführung nutzten, in der Hoffnung, dass sich das Ergebnis verändert. Um diese Charakteristik der Durchführungsphase herauszustellen, wurden die gesamten Transkripte wiederholt betrachtet. Dies war notwendig, da in den Codelines nur die Dauer der Phase zu erkennen ist und keine Rückschlüsse auf die Anzahl der Durchführungen eines Programms möglich waren.

3.4 Zusammenfassung und Diskussion

Zur Beantwortung der Forschungsfrage „*Welche Gemeinsamkeiten und Unterschiede weisen die Prozesse Physical Computing und die wissenschaftliche Erkenntnisgewinnung auf? Kann Physical Computing als Arbeitsweise der Erkenntnisgewinnung in der Informatik genutzt werden?*“ muss zunächst das theoretische Modell des Physical-Computing-Prozesses empirisch gestützt werden. Die beschriebenen Phasen von O'Sullivan und Igoe (2004), Banzi (2011) und Okita (2014) konnten in den Studien wiedergefunden und das Prozessmodell damit empirisch untermauert werden. Geringe Adaptionen wurden an dem Modell vorgenommen, da eine Teilphase bislang vernachlässigt wurde bzw. die empirischen Daten dem in der Literatur beschriebenen Prozess widersprachen. Es wurde beobachtet, dass das *Zergliedern von Problemen in Teilprobleme* zumeist als Teil der Evaluationsphase stattfindet und nachdem bereits Schwierigkeiten aufgetreten sind. Eine weitere Teilphase wurde bislang in der Literatur vernachlässigt – es handelt sich um die *Identifikation von Unsicherheiten und Fehlern*, der durch die Kommunikation mit der Umgebung durch Sensorik eine wichtige Rolle zukommt. Ein möglicher Ansatz für weitere Forschung in diesem Bereich bezieht sich explizit auf die Teilphase der Identifikation von Unsicherheiten und Fehlern. Es ist möglich, dass die Schaffung von Verständnis über Messunsicherheiten und auftretende Fehler, die insbesondere durch Sensoren bedingt sind, den Physical-Computing-Prozess maßgeblich vereinfachen.
Wesentliche Unterschiede der Prozesse sind in ihrer Zielsetzung zu finden. Im Physical-Computing-Prozess wird das Ziel ähnlich beschrieben, wie in den Ingenieurswissenschaften üblich. In beiden Domänen wird ein gezielter Zustand angestrebt, der das Leben vereinfachen soll und dafür wird zumeist ein Produkt angefertigt. Nichtsdestotrotz werden in diesem Prozess Erkenntnisse über informatische und ggf. naturwissenschaftliche Zusammenhänge deutlich, sowie der Problemlöseprozess von WissenschaftlerInnen, um Erkenntnisse und Produkte zu generieren. In den Naturwissenschaften steht neben dem Erwerb von wissenschaftlichen Methoden und Kompetenzen zur Erkenntnisgewinnung die Untersuchung eines bestehenden natürlichen Phänomens im Vordergrund. Die Evaluationsphase des Physical-Computing-Prozesses scheint weniger strukturiert zu sein, was die Analyse von aufgenommenen Daten erschwert. Die Trennung aus der wissenschaftlichen Erkenntnisgewinnungsforschung in Aufbereitung, Verarbeitung und Interpretation von Daten im Physical Computing könnte in einigen Anwendungsfällen eine positive Wirkung auf die gesamte Evaluation haben. Prinzipiell sollte bei der Betrachtung des Physical-Computing-Prozesses darauf geachtet werden, dass Abstufungen zu dessen Überschneidung zum Prozess der wissenschaftlichen Erkenntnisgewinnung existieren. Handelt es sich z. B. um Aufgaben, bei denen mit diversen analogen Sensoren gearbeitet wird, scheint die Überschneidung mit dem physikalischen Experiment größer zu sein als werden lediglich die Zustände *on* oder *off* unterschieden. Um diese Hypothese zu stützen, müssen jedoch systematische Untersuchungen mit verschiedenen Physical-Computing-Geräten so-

wie unterschiedlichen Aufgabenstellungen durchgeführt werden.
Gemeinsamkeiten der Prozesse sind in allen Phasen zu erkennen. Der Planungsprozess, die Konstruktions-/Implementations- sowie die Durchführungsphase teilen viele Inhalte miteinander, obwohl die verwendeten Geräte und durchgeführten Aktionen innerhalb der Phasen Unterschiede aufweisen. Der Verlauf des Physical-Computing-Prozesses weist ähnliche Merkmale auf wie bereits die von Patzwaldt et al. (2014) identifizierten Muster, d. h. die Evaluationsphasen wurden teilweise verkürzt oder übersprungen und ein iterativer Verlauf von Planung, Durchführung und Evaluation beobachtet. Darüber hinaus zeichnete sich bei Teilaufgaben zur Durchführung ein oszillierender Prozess zur Planung (vgl. der Implementation im Physical-Computing-Prozess) ab. In dieser Publikation schlussfolgern die AutorInnen, dass durch die Konstruktion ihrer Experimentieraufgaben, der Verlauf des Erkenntnisgewinnungsprozesses unterstützt werden konnte. Ob das Gleiche auch für den Physical-Computing-Prozess gilt, ist wahrscheinlich, jedoch bislang nicht empirisch abgesichert.
Die Untersuchung der Ähnlichkeit beider Prozesse wurde aus der Literatur heraus motiviert. Anhand der empirischen Überprüfung des Physical-Computing-Prozesses kann die Ähnlichkeit der Prozesse durch Daten gestützt werden. Es wurden ähnliche Phasen bzw. Teilphasen im Physical-Computing-Prozess wiedergefunden, welche bereits in der wissenschaftlichen Erkenntnisgewinnung beschrieben wurden. Insbesondere sind alle Phasen des wissenschaftlichen Erkenntnisgewinnungsprozesses im Physical-Computing-Prozess wiedergefunden worden und es wurden keine Phasen gefunden, die nicht in diese Phasen einsortiert werden konnten. Es kann geschlussfolgert werden, dass die Prozesse parallel ablaufen, wie es bereits für den Design- und wissenschaftlichen Erkenntnisgewinnungsprozess von National Research Council (1996) beschrieben wurde. Aufgrund der Parallelität beider Prozesse, ist es naheliegend, dass der zweite Teil der Forschungsfrage bestätigt werden kann.

Herausgearbeitete Gemeinsamkeiten und Unterschiede des Physical-Computing-Prozesses und des wissenschaftlichen Erkenntnisgewinnungsprozesses werden an dieser Stelle kurz zusammengefasst.
Gemeinsamkeiten:

- ähnliche Aktivitäten innerhalb der Phasen und Teilphasen der Prozesse,
- iterativer Verlauf der Prozesse und von Teilprozessen,
- Phasen und Teilphasen werden teilweise übersprungen oder in abweichender Reihenfolge durchlaufen,
- Messunsicherheiten und Fehler scheinen eine wichtige Rolle in den Prozessen zu zukommen.

Unterschiede:

- Ausgangspunkt/Motivation zum Beginn der Prozesse,
- Implementationsphase im Physical-Computing-Prozess umfasst zusätzlich die Implementation von Software (neben der Konstruktion von Hardware),
- über den oszillierenden Verlauf von Phasen hinaus wurden diverse Übergänge von Phasen im Physical-Computing-Prozess festgestellt.

Implikationen für die Schulpraxis sind, dass interdisziplinäre Unterrichtseinheiten mit Physical-Computing-Geräten vielversprechend sind, um beschriebene Problemlöseprozesse zu fördern. Es gilt zu überprüfen, ob bereits existierende Unterrichtseinheiten aus der Informatik bzw. den Naturwissenschaften genutzt werden können, um interdisziplinäre MINT-Kontexte zu thematisieren und Kompetenzen zu fördern. Es kann z. B. in der Informatik verstärkt auf die Funktionsweise von Sensoren, das Messen und die Konstruktion von Experimenten eingegangen werden. Im naturwissenschaftlichen Unterricht hingegen kann auf die benutzten Messgeräte (sofern Physical-Computing-Geräte verwendet werden) eingegangen werden und wie diese Geräte Daten verarbeiten.
Bei der Entwicklung von Physical-Computing-Aufgaben für den Unterricht sollten Teilaufgaben formuliert werden, die die SuS durch die Phasen des Physical-Computing-Prozesses leiten.
Für die Forschung innerhalb der MINT-Didaktiken bedeuten diese Ergebnisse, dass auf ein gemeinsames Modell für den MINT-Bereich zurückgegriffen werden kann. Die Modelle des wissenschaftlichen Erkenntnisgewinnungsprozesses und des Physical-Computing-Prozesses teilen wichtige Gemeinsamkeiten bei dem Verlauf der Prozesse und können ähnliche Thematiken und gleiche Geräte (wie Mikrocontroller und Mini-Computer) aufgreifen. Auf dieser Basis können interdisziplinäre und fachspezifische Aspekte der Problemlöseprozesse in einem Modell vereinigt werden.
Anhand der prozessbezogenen Überschneidungen stellt sich die Frage, inwieweit bereits existierende Ergebnisse der Erkenntnisgewinnungsforschung für die Informatikdidaktik genutzt und adaptiert werden können, z. B. Probleme der SuS während des Problemlöseprozesses und mögliche Unterstützungsmöglichkeiten. Darüber hinaus ist die Entwicklung von validierten Messinstrumenten im Bereich der wissenschaftlichen Erkenntnisgewinnung in den Naturwissenschaften bereits Bestandteil der Forschung. Es wurde keine Evidenz gefunden, dass ähnliche validierte Instrumente in der Physical-Computing-Forschung bislang eine Rolle spielten. Eine weitere Forschungslücke befindet sich in der Frage der Geräteabhängigkeit in dem Problemlöseprozess. Die Studien 1 - 4 wurden mit LEGO Mindstorms-Robotern und Arduino-Mikrocontrollern durchgeführt. Im Ablauf des Physical-Computing-Prozesses (anhand von Codelines) konnte lediglich die Tendenz gefunden werden, dass die Länge der Implementationsphasen des Physical-Computing-Prozesses

mit den verwendeten Geräten variiert. Diese Tendenz geht aus den Codelines hervor, wobei weitere Forschung für eine Absicherung notwendig ist. Es gilt, weitere prozessbezogene Unterschiede der Geräte in Abhängigkeit zu ihrer Komplexität zu untersuchen, um den Physical-Computing-Prozess detailliert abbilden zu können.
Die aufgezeigten Ergebnisse sind limitiert bezüglich der in den Studien verwendeten Physical-Computing-Geräte, Aufgabenstellungen und teilnehmenden Altersgruppen.
Die Ergebnisse dieses Kapitels können als Grundlage für weitere fachdidaktische Forschung genutzt werden. Sie bieten ein Modell, das den Problemlöseprozess von SuS während Physical-Computing-Aufgaben beschreibt. Zukünftig könnte es unterstützend als Modell wirken, um z. B. gezielt Lern- und Lehrmaterial für den Physical-Computing-Unterricht zu konstruieren.

Diskussion des Einflusses von der Hilfestellung *4-Phasen analog*

Zur Beantwortung der 1. Forschungsfrage bekamen die SuS die Hilfestellung *4-Phasen analog* gegeben. In diesem Abschnitt soll der Einfluss dieser Hilfestellung auf den erhobenen Physical-Computing-Prozess und damit eine mögliche Verfälschung der Ergebnisse abgewogen werden.
Im Folgenden werden die Gruppen, die die Hilfestellung nutzten, denen gegenübergestellt, die die Arbeitsblätter (Hilfestellung *4-Phasen analog*) nicht ausfüllten. Beide Gruppen der 1. Studie füllten auf dem Arbeitsblatt die Felder der Vorbereitungs- und Implementationsphase aus, wobei die Codelines erkennen lassen, dass das Ausfüllen nachträglich erfolgte. Der Bearbeitungsprozess der Aufgaben (siehe Codelines der Abbildung 3.27) verlief nicht parallel zu der Beschreibung der Arbeitsphasen auf den Arbeitsblättern (vgl. Anhang C.1 und C.2). So startete Gruppe AB z. B. nicht mit einer Vorbereitungsphase, sondern begann mit der Implementation bis hin zur Evaluation, bevor sie zum ersten Mal ihr Vorgehen explizit planten. Ebenfalls ist kein idealtypischer zyklischer Verlauf des Prozesses zu erkennen, sondern das Überspringen von Phasen. Sie ähnelt der 2. Studie, in der keine Gruppe die Hilfestellung nutzte. In der 4. Studie nutze die Gruppe EF die Hilfestellung *4-Phasen analog* (vgl. Abbildung 3.24 und Anhang C.3). Auch diese Gruppe füllte lediglich ein Feld der Hilfestellung aus. Zusammenfassend zeigen sich auf der Basis der Studiendaten keine Auffälligkeiten dahingehend, dass die Hilfestellung *4-Phasen analog* den Prozess beeinflusste. Es ist festzuhalten, dass die Prozesse in der Phasenlänge und der Häufigkeit des Phasenüberspringens unterschiedlich sind, wobei das für alle Gruppen gilt. Es sind keine Auffälligkeiten bei den Gruppen zu finden, die die Hilfestellung *4-Phasen analog* verwendeten im Gegensatz zu den Gruppen, die sie zu keinem Zeitpunkt nutzten (beispielsweise häufige Vorbereitungsphasen, lange Implementationsphasen und Auslassen von Evaluationsphasen).
Aufgrund des Designs der Hilfestellung *4-Phasen analog* sollte sie lediglich als Stimulus dienen, um den SuS den Einstieg in die Aufgabenbearbeitung zu erleichtern und die SuS

nicht dauerhaft durch den Ablauf des Physical-Computing-Prozesses zu leiten. Deshalb wurden die untersuchten Kategorien des Physical-Computing-Prozesses nicht konkret benannt, sondern durch Fragen umschrieben. Darüber hinaus wurden Doppelfragen gestellt, die zu einer großen kognitiven Beanspruchung führen können und die SuS ggf. nicht wissen, welche Frage sie beantworten sollen (vgl. Thielsch et al., 2012). Für eine explizite Beeinflussung des Prozesses zeigen Ergebnisse der Inquiry-Forschung, dass insbesondere direkte Hilfestellungen benötigt werden (vgl. Kirschner et al., 2006) und SuS Übungszeit für den Umgang mit der Hilfestellung benötigen (vgl. Arnold et al., 2017).
Abgeleitet von den Daten der Studien 1 - 4 und den aufgezeigten Forschungsergebnissen zur Konstruktion von Hilfestellungen ist zu schlussfolgern, dass Hilfestellung *4-Phasen analog* den Physical-Computing-Prozess nur marginal beeinflusst haben kann.

Kapitel 4

Probleme während des Physical-Computing-Prozesses

In Kapitel 2 wurden bereits das Forschungsfeld Physical Computing vorgestellt und verschiedene Problemstellungen aufgezeigt. Im Rahmen der 1. Forschungsfrage ist der Problemlöseprozess bei der Arbeit mit Physical-Computing-Geräten detailliert beschrieben worden (vgl. Kapitel 3). Es konnte festgestellt werden, dass dieser Prozess diverse Ähnlichkeiten zum Prozess der wissenschaftlichen Erkenntnisgewinnung aufweist. Basierend darauf gewinnt eine weitere im Literaturreview identifizierte Forschungslücke an Relevanz. Wie bereits in Kapitel 2.2.1 dargelegt, ist der Umgang mit Problemen bzw. die Vielfalt von auftretenden Problemen innerhalb des Prozesses der wissenschaftlichen Erkenntnisgewinnung eine große Herausforderung, um den Prozess erfolgreich zu bewältigen. Es ist naheliegend, dass SuS bei der Arbeit mit Physical-Computing-Geräten ähnliche Probleme haben, wie sie beim Experimentieren in den Naturwissenschaften beschrieben wurden. Um den Problemlöseprozess gezielt unterstützen zu können, muss zunächst geklärt werden, mit welchen konkreten Problemen die SuS umgehen müssen und worin die Ursachen für diese Probleme liegen. Diese Analyse innerhalb des Physical-Computing-Prozesses ist unabdingbar, da durch die Verwendung von technischen Geräten und insbesondere Sensoren Probleme auftreten. Diese Geräte sind per se fehleranfällig. Zunächst muss auf einer wissenschaftlichen Ebene das Auftreten von Problemen untersucht werden, um sie an Lehrkräfte sowie SuS zurückzumelden und den Lernprozess zu unterstützen. Masnick und Klahr (2003, S. 67) motivieren ihre Untersuchung von Fehlern der SuS wie folgt: „Indeed, the acquisition of a deep understanding about different types of error and procedures for dealing with them constitutes an important part of the professional training of scientists in all disciplines.“ Die AutorInnen verdeutlichen, dass die Entwicklung eines Verständnisses über Fehlertypen ein wichtiger Bestandteil für wissenschaftliches Arbeiten ist. Darüber hinaus bildet das Wissen über mögliche Fehler und Probleme eine Grundlage für Hilfestellungen für den Lernprozess. Eine Kategorisierung kann für die Identifikation von Fehlern und Problemen hilfreich sein. Das Ziel dieses Kapitels besteht darin, ein

möglichst allgemeines Modell für Probleme und deren Ursachen im Physical-Computing-Prozess zu erstellen, das auf verschiedene Physical-Computing-Geräte anwendbar ist.
Die aufgezeigte Forschungslücke wird durch die folgende Fragestellung adressiert:

Welche Probleme treten bei der Interaktion mit Physical-Computing-Geräten auf und wie können diese kategorisiert werden?

Bereits angesprochene Terminologien werden in diesem Kapitel wie folgt verwendet: *Fehler* können technischer oder menschlicher Natur sein, die in konkreten Situationen sichtbar werden können. Fehler führen zu *Problemen*, da sie den Weg zur Erreichung eines angestrebten Zielzustands erschweren (vgl. Newell und Simon, 1976). Probleme wiederum haben eine Ursache, die als Problemursache bezeichnet wird.
Gemäß des DBR-Ansatzes wird zuerst ein Literaturreview vorgenommen, dass Beschreibungen von Problemen der SuS während der Interaktion mit Physical-Computing-Geräten beinhaltet (vgl. Kapitel 4.1). Dabei wird auf die Physical-Computing-Literatur eingegangen, in der implizit auf Probleme der SuS während des Physical-Computing-Prozesses hingewiesen wird. Dieser Abschnitt entspricht der Phase Analysis & Exploration II im Gesamtprozess. Anschließend wird das methodische Vorgehen vorgestellt, mit der die Forschungsfrage empirisch untersucht wird (vgl. Kapitel 4.2, entspricht der Phase Design & Construction II). In Kapitel 4.3 (Evaluation & Reflection II) werden die in der Literatur beschriebenen Kategorien von Problemursachen anhand induktiver Erkenntnisse detaillierter beschrieben (Abschnitt 4.3.1). Hinweise auf weitere Kategorien werden beispielhaft in Abschnitt 4.3.2 dargestellt. Nach der qualitativen Analyse erfolgt eine Visualisierung quantitativer Daten, die lediglich Tendenzen aufzeigen, wie die Probleme in diesen Studien verteilt waren.

4.1 Stand der Forschung zu Problemen im Physical-Computing-Prozess

Bislang fokussiert sich die Forschung vorwiegend auf ausgewählte Physical-Computing-Geräte und ihren Einsatz im Unterricht sowie ihre Auswirkung auf die Motivation von SuS. Erste Hinweise auf Problemursachen im Physical-Computing-Prozess lassen sich durch die Charakterisierung des Physical-Computing-Begriffs ableiten, welche die Kommunikation der physischen Welt mit der virtuellen Welt des Computers umfasst (vgl. O'Sullivan und Igoe, 2004). Die physische Welt wird von O'Sullivan und Igoe als *Umgebung* abgeleitet und die virtuelle Welt des Computers besteht aus der *Software*. Die *Hardware* ist ebenfalls ein Bestandteil der physischen Welt, wobei sie die Verbindung beider Welten ermöglicht. Diese Komponenten finden sich ebenfalls in weiteren Beschreibungen des Physical-Computing-Begriffs bzw. von Physical-Computing-Aktivitäten wieder (vgl. Przybylla und Romeike, 2014a). In der Physical-Computing-Forschung werden bisher selten spezifische Probleme der Lernenden betrachtet, so dass in diesem Abschnitt bereits existierende Ansätze und

Hinweise auf Probleme und Problemursachen aufgezeigt werden sollen.
Okita weist auf ein Problem hin, das sie als *rekursives Feedback* bezeichnet. Dahinter verbirgt sich die Diskrepanz zwischen dem geschriebenen Programm (mit den damit verfolgten Intentionen) und den tatsächlichen Wirkungen des Programms z. B. in Form der Roboterbewegungen (Okita, 2014). Das grundlegende Problem besteht darin, dass das Programm am Computer geschrieben und anschließend auf den Roboter oder ein anderes Gerät zur Ausführung des Programms übertragen wird. Zumeist wird dabei sogar der Standort innerhalb der Umgebung gewechselt und die Lernenden haben keinen Einfluss mehr auf das Programm. Im Anschluss muss identifiziert werden, welcher Teil eines Programms zu welchem Verhalten des Roboters führte, so dass der Grund für ein eventuelles Fehlverhalten gefunden werden kann. Einerseits sind die geschriebenen Programme zum Teil sehr umfangreich, was die Zuordnung eines Fehlers zu einem Programmteil erschwert. Andererseits ist die zeitliche Verzögerung schwierig, da sich die SuS die Aktionen des Roboters genau merken müssen, um diese anschließend dem Programm zuzuordnen. In der Forschung der naturwissenschaftlichen Erkenntnisgewinnung wurde bereits gezeigt, dass es SuS schwer fällt ein zugrundeliegendes Problem zu verstehen und zu charakterisieren (vgl. Koppelt und Tiemann, 2009). Das Verständnis über ein Problem ist jedoch die Voraussetzung, um im Anschluss die Ursache innerhalb der Software oder anderen Systemkomponenten zu identifizieren und zu beheben.
Basierend auf den Erfahrungen bei der Durchführung eines E-Textilien-Projekts werden weitere Probleme beschrieben (Kafai et al., 2014b). Dabei wurde neben den Problemen im geschriebenen Code auch das Nähen von Schaltkreisen als Problemursache benannt. Kafai et al. beschrieben ihre Probleme mit „However, debugging e-textiles is a complex process, more so than debugging program code, because bugs can be caused by the code, circuit design, or crafting“ [S. 1:15]. Andere Ansätze verzichten hingegen gezielt auf die Überlappung mehrerer Konzepte und z. B. auf physikalische Kenntnisse, indem sie Hardware-Module und virtuelle IDEs für die Programmierung der Geräte nutzen (Katterfeldt et al., 2016). Hier werden als Problemursache die Software, die Konstruktion der Geräte und physikalische Kenntnisse identifiziert. Die manuellen Fähigkeiten wie das Nähen und Basteln werden von den AutorInnen als Herausforderungen angesprochen. In einer weiteren Studie wird Physical Computing interdisziplinär mit weiteren Fächern verbunden. Die Lernenden stellen explizit heraus, dass sie sich von dieser Art des Lernens als Hands-on-Aktivität sehr angesprochen fühlen, da sie kreativ tätig werden können. Die AutorInnen beschreiben in der Untersuchung, dass lediglich das Debuggen von Software große Probleme für sie bereitete (Cross et al., 2016).
In der Physical-Computing-Literatur wird eine breite Spanne an Problemen und Problemursachen benannt, ohne jedoch eine systematische Analyse vorzunehmen. Eine differenzierte Erörterung des Vorwissens der ProbandInnen fehlt teilweise und eine gezielte Beziehung von Problemursachen zu verwendeten Geräten und bearbeiteten Aufgabenstellungen wird bislang nur in Ansätzen beschrieben.

Im Rahmen der Forschung zur wissenschaftlichen Erkenntnisgewinnung wurde bereits eine Taxonomie von aufgetretenen Fehlern während des Inquiry-Prozesses vorgeschlagen. Im Anschluss wurde die Taxonomie genutzt um das Verständnis von SuS der 2. - 4. Klasse bezüglich Ursachen und Konsequenzen von Fehlern zu untersuchen (Masnick und Klahr, 2003). Die AutorInnen gliedern den Inquiry-Prozess in die Phasen: Entwicklung des Testdesigns (Variablen festlegen), Vorbereitung, Durchführung, Beurteilung der Auswirkungen und Auswertung. Als Fehlertypen benennen sie die Kategorien: Design-Fehler, Messfehler, Durchführungsfehler und Interpretationsfehler.
Design-Fehler werden beschrieben als „[they] occur 'in the head' rather than 'in the world'". Es handelt sich dabei um Probleme im Verständnis oder durch unzulängliches Wissen innerhalb einer Domäne. In der Entwicklung des Testdesigns äußern sich die Probleme z. B. durch die falsche Konzeptionalisierung und Operationalisierung von Variablen. Verglichen mit dem Physical-Computing-Prozess sind die beschriebenen Fehler der Vorbereitungsphase des Physical Computing zuzuordnen. Darin geht es nicht zwangsläufig um die Operationalisierung von Variablen, sondern es wird ein grober Plan entworfen, wie ein Problem gelöst werden kann. Es handelt sich ebenfalls um einen kognitiven Prozess, der anschließend als Algorithmus in der Software festgehalten wird, und dessen logische Fehler u. a. in der Software bemerkbar werden.
Masnick und Klahr beschreiben als Messfehler: 1.) die Nutzung von falschen Aufbauten und Messinstrumente (in der Vorbereitungsphase) und 2.) eine falsche Kalibrierung von Messgeräten (in der Durchführungsphase). Diese beschriebenen Fehler sind der Implementations- und Durchführungsphase des Physical-Computing-Prozesses zuzuordnen. Darin werden ebenfalls Konstruktionen vorgenommen und Messinstrumente kalibriert, bzw. muss mit Fehlern umgegangen werden, die aus der Kalibrierung von Sensoren resultieren. Da diese Fehler Hardwarekomponenten tangieren, können sie im Physical Computing zunächst dem Bereich Hardware zugeordnet werden.
Als Durchführungs- bzw. Interpretationsfehler können unerwartete, unbekannte und unbemerkte Einflüsse auf den Prozess auftreten. Diese werden der Phase der Durchführung des Experiments zugeordnet und beeinflussen den Ausgang des Experiments. Diese Fehler können zufällig auftreten, wie z. B. dass bei der mehrfachen Durchführung eines Experiments unterschiedliche Auswirkungen beobachtet werden können. Der Fehler kann offensichtlich sein oder unbemerkt bleiben. Anhand des folgenden Experiments werden diese Fehler beschrieben: ein Ball wird von einer Rampe herunter gerollt und es soll der Punkt bestimmt werden, an dem der Ball zum Stillstand kommt. Ein offensichtlicher Fehler wäre, wenn der Ball die Wand der Rampe berührt und dadurch die Bewegungsbahn verändert wird. Unbemerkte Fehlern können z. B. darin bestehen, dass die Oberfläche des Balls fehlerhaft ist, was ebenfalls die Bewegungsbahn beeinflusst. Die beschriebenen Einflüsse sind ebenfalls im Physical-Computing-Prozess wiederzufinden und können den Phasen der Durchführung und Evaluation zugeordnet werden, wobei sie vor allem die Umgebung im Physical Computing adressieren.

Zusammengefasst werden die folgenden Problemursachen von den AutorInnen explizit für den Bereich Physical Computing genannt: Software (der Programmcode, den die SuS erzeugen), physikalische Grundlagen und die Konstruktion von Hardware. Das Basteln bzw. die Handarbeit wird an dieser Stelle vernachlässigt, da es nur auf einige Physical-Computing-Geräte zutrifft und nicht verallgemeinerbar ist. Abgeleitet von der Definition von Physical Computing spielen Umgebungseinflüsse neben der Software, der Hardware und dem physikalischen Grundlagen eine Rolle in einem Physical-Computing-System. Die Umgebungseinflüsse werden im Folgenden ebenfalls betrachtet. Prinzipiell sind die hier aufgezählten Problemursachen jedoch zu allgemein, um sie an SuS oder Lehrkräfte zurück zu melden. Aus diesem Grund sollten die Problemursachen genauer untersucht werden.

4.2 Untersuchungsmethode

Diese Forschungsfrage wird anhand von Daten der ersten experimentellen Studien untersucht (vgl. Kapitel 3). Die erhobenen Videodaten wurde unter der Nutzung eines weiteren Codiermanuals (vgl. Tabelle 4.1) erneut codiert. Sofern mindestens eine der folgenden Voraussetzungen vorhanden war, wurden Aktionen und Situationen als problematisch eingestuft, d. h. ein Problem liegt vor:

1. Die SuS fragten die Lehrkraft nach Hilfe, da sie nicht wussten, wie sie der Lösung der Aufgabe näher kommen.

2. Die Lehrkraft entschied einzugreifen, da die SuS für mehrere Minuten an einer Stelle im Problemlöseprozess stagnierten.

3. Die transkribierende Person bemerkte eine Situation, in der die SuS Fehler machten, die die SuS bemerkten und nicht als Problem einstuften, oder die SuS bemerkten die Fehler nicht und das führte zu Problemen.

Das Codiermanual (Tab. 4.1) umfasst als deduktive Kategorien Hardware, Software und Umgebung, die aus den in der Literatur beschriebene Komponenten eines Physical-Computing-Systems abgeleitet wurden. Die Beschreibung, dass es sich im Bereich der Software um das selbst geschriebene Programm der Schülerinnen und Schüler handelt, wurde deduktiv hergeleitet. Das Problem, eine zielgerichtete Konstruktion zu finden, wurde ebenfalls bereits im Bereich der Hardware beschrieben. Physikalische Grundlagen werden in der Literatur ohne detaillierte Erklärungen genannt und deshalb als weitere deduktive Kategorie aufgenommen.
Die Offenheit für weitere (induktive) Kategorien bestand weiterhin, so dass diese anschließend mit ihrer Definition im Codiermanual festgehalten bzw. bestehende Kategorien detaillierter beschrieben wurden.
Es wurde zunächst ein vorläufiges, deduktives Modell von den Problemursachen des

Physical-Computing-Prozesses erstellt und anschließend induktiv durch Daten aus folgenden Studien ergänzt und adaptiert.
Die gefundenen Probleme wurden durch einen zweiten Codierer überprüft. Dieser Codierer bekam 10 % der Transkripte, in denen bereits gefundene Probleme markiert wurden. Der zweite Codierzyklus bestand darin, die Probleme den vorgegebenen Problemursachen zuzuordnen. Es konnten alle Problemursachen (deduktive und induktive) durch den zweiten Codierer wiedergefunden und bestätigt werden. Lediglich bei Problemen mit mehrdeutigen Problemursachen ordnete dieser Codierer einen Fehler tendenziell mehr Problemursachen zu als der erste Codierer. Für das abgeleitete Modell ist jedoch besonders wichtig, dass alle Kategorien wiedergefunden werden konnten, da bei vielen Problemen keine Trennschärfe existiert. Es wurde auf die Berechnung einer Interrater-Reliabilität verzichtet, da für diese Untersuchung die Überprüfung der aufgestellten Taxonomie im Vordergrund stand und abgesichert werden sollte. Es sollte sichergestellt werden, dass für alle aufgestellten Kategorien Beispiele zu finden sind. Die eindeutige Zuordnung eines Problems zu genau einer Kategorie der Taxonomie wird zu diesem Zeitpunkt nicht angestrebt.

Codiermanual

Zur Beantwortung der 2. Forschungsfrage, werden in den Studien entstandene Transkripte (zugehörige Studien aus dem Studiendesign entnehmbar auf Seite 16) auf die folgenden Codes untersucht. Diese sind deduktiv und induktiv erzeugt worden.

Tabelle 4.1: Codiermanual zur Forschungsfrage 2 (Probleme und Problemursachen im Physical Computing)

Name (Abkürzung)	**Definition**	**Beispiel**
Software (SW)	Alle Probleme, die beim Programmieren des Roboters auftreten, bzw. Probleme, die auf das erstellte Programm zurückzuführen sind; Probleme beim Umgang mit bestehender Software, wie der Programmierumgebung oder der Firmware des Geräts.	SuS verstehen den Unterschied von Schleifen und Bedingungen nicht.
Hardware (HW)	Es handelt sich um eine Fehlfunktion (Kalibrierung, Ausfall) von Sensoren oder Aktuatoren. Die Konstruktion des gesamten Geräts (z. B. Kabelführung) ist ein weiterer Aspekt, wobei die Hardware nicht selbstständig konstruiert worden sein muss.	Ein Rad kann sich aufgrund der Konstruktion nicht einwandfrei drehen.

Umgebung (EN)	Einflüsse, die von außen auf den Roboter einwirken, erzeugen ein Problem. Dieses kann durch natürliche Einflüsse oder das Eingreifen von Menschen ausgelöst sein.	Lichtverhältnisse verändern sich, ggf. durch ungeeignete Positionierung der SuS im Raum.
Mathematische/ physikalische Grundlagen (MP)	Vorkenntnisse oder Konzepte aus der Mathematik oder Physik fehlen bei den SuS bzw. existieren Schwächen, die Probleme hervorrufen.	SuS setzen den Grenzwert auf genau einen Wert anstatt auf ein Intervall; Kenntnisse über Operatoren.
Software/ Hardware (SW/HW)	Es handelt sich um eine Überschneidung der Bereiche HW und SW. Das Problem liegt in beiden Bereichen und kann zumeist durch Veränderungen in einem der beiden Bereiche behoben werden.	Der Roboter wurde auf eine Weise programmiert (SW), die für die Konstruktion (HW) ungeeignet ist.
Software/ Umgebung (SW/EN)	Es handelt sich um eine Überschneidung der Bereiche SW und EN. Das Problem liegt in beiden Bereichen und kann zumeist durch Veränderungen in einem der beiden Bereiche behoben werden.	Ein Mensch bewegt den fahrenden Roboter (EN), damit Schwachstellen der Software ausgeglichen werden und trotzdem das gewünschte Ergebnis erzielt wird.
Hardware/ Umgebung (HW/EN)	Es handelt sich um eine Überschneidung der Bereiche HW und EN. Das Problem liegt in beiden Bereichen und kann zumeist durch Veränderungen in einem der beiden Bereiche behoben werden.	Ein Sensor wurde so konstruiert, dass er regelmäßig versehentlich abgedeckt wird.
Gesamtsystem (SY)	Es handelt sich um eine Überschneidung von mehr als zwei der Problemursachen gleichzeitig, d. h. ein Problem liegt in > 2 Problemursachen. Einflüsse von SW, HW und EN beeinflussen das Gesamtsystem gleichzeitig.	Der Roboter kann nicht exakt geradeaus fahren und sich ebenfalls nicht um genau 90 Grad drehen.

4.3 Ergebnisse

Im Folgenden werden die untersuchten Hauptproblemursachen beschrieben. Sie wurden deduktiv erstellt und auf Grundlage der erhobenen Daten detaillierter beschrieben (vgl. Kapitel 4.3.1). Induktiv abgeleitet wurden die mehrdeutigen Problemursachen, die in Kapitel 4.3.2 beschrieben werden.

4.3.1 Beschreibung der Hauptproblemursachen

Bei der qualitativen Analyse wurden vor allem die Komponenten eines Physical-Computing-Systems – Software, Hardware und Umgebung – als Problemursachen wiedergefunden. Hinzu kam eine vierte Hauptkategorie, die mathematische und physikalische Grundlagen beinhaltet. Im Folgenden werden zunächst die Hauptproblemursachen und ihre Unterkategorien detailliert beschrieben.

Hardware umfasst alle physischen Komponenten, die an einem Physical-Computing-Gerät montiert sind bzw. das Gerät selbst. In der Robotik ist es z. B. der Roboter bzw. einzelne Komponenten wie Sensoren und Aktuatoren, die bei LEGO Mindstorms modular verwendet werden können. Bei der Arbeit mit Mikrocontrollern beinhaltet diese Kategorie den Mikrocontroller sowie die dazugehörige Peripherie an Sensoren, Aktuatoren, Steckbrettern, Kabeln und weitere Materialien zur Konstruktion. Diese Kategorie bezieht sich ebenso auf die Verwendung der genannten Hardwarekomponenten, das heißt wie der Aufbau bzw. die Konstruktion gestaltet ist. Das kann die Wahl der Position von Sensoren und Aktuatoren betreffen. Des Weiteren kann die Stabilität der Konstruktion ein Faktor sein, damit das Gerät seine Funktion erfüllen kann. Auch die Funktionalität der Hardware zählt zu diesem Bereich, wie z. B. Messgrenzen von Sensoren oder die Genauigkeit von Motoren. Die Kategorie Hardware kann in die folgenden drei Unterkategorien eingeteilt werden:

1a Die Konstruktion ist ungeeignet und verursacht deshalb Probleme, z. B. die Kabelführung vom Mikrocontroller zu den Sensoren/Aktuatoren. Einige SuS führten Verbindungskabel direkt vor dem Ultraschallsensor entlang oder befestigten den Sensor nicht parallel zum Boden, was die maximale Messweite des Sensors negativ beeinträchtigte. (bereits deduktiv hergeleitet)

1b Einzelne Bestandteile der Hardware funktionieren fehlerhaft, z. B. eine falsche Kalibrierung oder Signalübersetzung von Sensoren oder Motoren. Das äußerte sich u. a. darin, dass bei dem gleichen verwendeten Programm unterschiedliche Motoren keine identische Rotation erzeugten und sich auch Entfernungen voneinander unterscheiden, die durch verschiedenen Ultraschallsensoren gemessen wurden.

1c Einzelne Bestandteile der Hardware, z. B. Sensoren oder Motoren funktionieren nicht/sind defekt, d. h. ein Sensor konnte gar nicht ausgelesen bzw. ein Aktuator konnte nicht angesteuert werden.

Software umfasst sowohl Programme, die das Physical-Computing-Gerät ansteuern als auch die zur Erstellung von Programmen notwendig sind. Im Fokus steht das Programm, dass die SuS schreiben, um das Gerät anzusteuern. Dabei wird zunächst keine Unterscheidung vorgenommen, auf welcher Programmiersprache es basiert oder ob sie textbasiert oder grafisch ist. Des Weiteren kann die Software bereits vorliegen und nur als Anwendung für die NutzerInnen dienen. Das wäre z. B. die Programmierumgebung, mit der die SuS das Gerät programmieren. Ebenfalls kann es die Firmware eines Roboters bezeichnen. Die SuS interagieren lediglich mit diesem Softwaresystem, wobei eine falsche Bedienung das Physical-Computing-System ebenfalls beeinträchtigen kann.

2a Der Programmcode, der von den SuS produziert wurde, ist fehlerhaft. Dies betrifft auch gänzlich fehlende Kontrollstrukturen zur Wiederholung des Programms, z. B. Schleifen. Das war der Fall, wenn die SuS die gemessene Entfernung des Sensors durch eine bedingte Anweisung mit einer Aktion verknüpften, diese Anweisung jedoch nur einmal wiederholt wurde, da die Schleife fehlte. Die Wahl von kopf- oder fußgesteuerten Schleifen war ebenfalls schwer für einige SuS. (bereits deduktiv hergeleitet)

2b Probleme mit der Programmierumgebung, auf der die SuS arbeiten und ihr Programm entwickeln. Diese Kategorie betrifft Probleme beim Umgang mit der Programmierumgebung z. B. das Verständnis bezüglich der Funktion von verschiedenen Programmblöcken. Die SuS hatten Probleme diverse „Motorblöcke“ geeignet einzusetzen, da die Unterschiede in der verwendeten Software nicht beschrieben waren. Teilweise waren Blöcke missverständlich beschrieben und suggerierten ein abweichendes Verhalten von den tatsächlichen Auswirkungen.

2c Die Software (Firmware) auf dem Roboter ist ebenfalls ein Spezialfall der Kategorie Software. Dieser Fall bezieht sich insbesondere auf Usability-Aspekte der Firmware, mit der beispielsweise bei der Auswahl von Funktionen (wie dem Starten des geschriebenen Programms) auf dem Roboter interagiert wird. So muss z. B. ein konkretes auf dem Roboter gespeichertes Programm ausgewählt und gestartet werden. Dadurch ist es möglich, versehentlich ein anderes Programm zu starten. Hinzu kam, dass mehrere Programme mit dem gleichen Namen benannt werden konnten, was die Auswahl erschwerte.

Umgebung umfasst alle äußeren Einflüsse auf das Physical-Computing-System. Diese können sich auf Einflüsse aus der Umwelt beziehen, die durch Sensoren messbar sind. Bei Umgebungseinflüssen werden zumeist Veränderungen dieser Einflüsse betrachtet. Es werden Daten in einer bestimmten Situation aufgenommen, z. B. Helligkeitswerte, die jedoch natürlichen Veränderungen unterliegen und somit rapide wechseln können. Die Umgebung kann außerdem durch menschliche Einflüsse verändert werden, z. B. wird der Roboter bei

der Ausführung des Programms durch die SuS bewegt und unterliegt dadurch ebenfalls anderen Umweltbedingungen. Ein anderer menschlicher Faktor ist der unbewusste Eingriff in das Physical-Computing-System, wenn der Ultraschallsensor z. B. eine menschliche Hand misst und eine Reaktion daraufhin ausgelöst wird, ohne dass menschliches Eingreifen in dem Algorithmus vorgesehen wurde. Unterteilt wird der Bereich Umgebung wie folgt:

3a Natürliche Umgebungseinflüsse, z. B. wechselnde Lichtverhältnisse; Gegenstände, die in der Umgebung stehen. Dazu gehören ebenfalls Unebenheiten oder Staub auf dem Boden, was das Fahrverhalten von Rädern beeinflussen kann. Die verschiedenen Lichtquellen im Raum (durch Deckenlampen oder Sonneneinstrahlung) wurden selten von den SuS bedacht.

3b Umgebungseinflüsse, die auf das menschliche Handeln zurückzuführen sind, z. B. eine aktive Veränderung der Umgebung. Einige SuS veränderten die Startposition des Roboters in verschiedenen Tests. Da Gegenstände, mit denen interagiert wurde, eine andere Entfernung aufwiesen als zuvor, führte dies zu unterschiedlichen Testbedingungen. Teilweise kam es vor, dass die SuS während der Programmausführung unbeabsichtigt von dem Ultraschallsensor gemessen und somit als Hindernis erkannt wurden. Dadurch veränderte der Roboter die Fahrtrichtung, was bedeutet, dass durch die SuS in das System eingegriffen wurde. Sie griffen ebenfalls beabsichtigt in die Fahrtrichtung des Roboters ein, wenn sie sich davon versprachen dem Zielzustand näher zu kommen.

Mathematische/Physikalische Grundlagen bezeichnet ein allgemein gehaltenes Gebiet, das Vorstellungen, Wissen und Kompetenzen in Mathematik und Physik beinhaltet. Das umfasst z. B. den Aufbau eines Experiments, die methodische Durchführung eines Experiments und die Berücksichtigung von Störfaktoren. In der Auswertung von Experimenten benötigen die SuS Kompetenzen um adäquate Schlüsse aus den erhobenen Daten ziehen zu können. Ebenfalls benötigen die SuS Wissen über die Verwendung von Terminologien, Formeln (bspw. Dreisatz, Mittelwerte) und Vergleichsoperatoren, die in den mathematisch-naturwissenschaftlichen Bereich einzuordnen sind. In diese Kategorie einzuordnen sind auch Konzepte, wie z. B. die Suche nach einem *wahren Wert*, wenn Messungen vorgenommen werden bzw. Kenntnisse über Messunsicherheiten. Somit können die folgenden Unterkategorien beschrieben werden:

4a Unsicherheiten der SuS in Bezug auf die Wahl von Intervallgrenzen und geeigneter Vergleichsoperatoren, um diese zu testen. Insbesondere verwechselten die SuS die Operatoren $<$ und $>$. Zudem wurde bei $\geq$ und $\leq$ der Fall nicht berücksichtigt, dass der zu vergleichende Wert mit dem Schwellwert übereinstimmt. Dies führte u. a. zu falsch aufgebauten Fallunterscheidungen.

4b Kenntnisse über die physikalische Funktion von Sensoren, z. B. die Funktionsweise des Ultraschallsensors. Dabei kam es zu Fehlern, wenn die SuS vernachlässigten, dass beide Seiten des Sensors (Sender und Empfänger) benötigt werden, um eine Messung durchzuführen. Der funktionale Unterschied von analogen und digitalen Signalen bereitete ebenfalls Probleme.

4c Schwierigkeiten der SuS beim Aufbau eines physikalischen Experiments, z. B. zur Ermittlung von Grenzen des Ultraschallsensors. Dabei nutzten die SuS oftmals ihre Hand vor dem Sensor, um festzustellen, ob diese erkannt wird. Auch beim Testen von unterschiedlichen Lichtbedingungen mit dem Lichtsensor traten ähnliche Probleme auf. Die SuS verdeckten den Sensor, indem sie ihre Hand darüber hielten, was eine ungenaue und fehleranfällige Vorgehensweise ist.

4d Wissen über Schwellwerte und wie diese abhängig von der Aufgabenstellung günstig gewählt werden. Alle SuS legten bei der Verwendung von bedingten Anweisungen einen Schwellwert fest. Dabei wählten sie nur teilweise einen Wertebereich (durch Operatoren) als Bedingung aus. Andere SuS legten einen konkreten Schwellwert fest, was ein Problem sein kann. Sie benötigen dafür Wissen, um die Genauigkeit und die Abtastrate von Sensoren zu bewerten und welche Konsequenzen es für Fallunterscheidungen haben kann. In Abhängigkeit von der Abtastrate eines Sensors und der Fahrgeschwindigkeit eines Roboters ist es möglich, dass einige Messwerte übersprungen werden. Das kann heißen, dass in einigen Fällen das Testen auf einen Wert durch ein geeignet großes Intervall ersetzt werden sollte.

4.3.2 Mehrdeutige Problemursachen

Bei der qualitativen Analyse wurden vor allem die Systemkomponenten Software, Hardware und Umgebung wiedergefunden und genauer beschrieben. Hinzu kam eine vierte Problemursache, die mathematische und physikalische Kenntnisse umfasst. Besonders interessant sind jedoch die gefundenen Überschneidungen der bisher aufgezeigten Problemursachen. In der Datenanalyse zeigt sich, dass viele Probleme nicht in einer Kategorie von Problemursachen einzuordnen sind, sondern gleichzeitig mindestens zwei Kategorien betreffen. Das erschwert es den SuS das genaue Problem zu identifizieren und Lösungsansätze zu entwickeln.
Die Überschneidungen zeigen sich in den Bereichen *Hardware/Software*, *Hardware/Umgebung* und *Software/Umgebung*. Sind alle dieser drei Systemkomponenten betroffen bzw. überschneiden sich, so wird im Folgenden von dem *System* als Problemursache gesprochen. Die Kategorie *mathematische/physikalische Grundlagen* weist ebenfalls Überschneidungen mit anderen Systemkomponenten auf. In diesem Zusammenhang werden diese jedoch nicht betrachtet, da es sich um Voraussetzungen für den Umgang mit Physical-Computing-Geräten handelt. Des Weiteren wird eine breite Datenbasis benötigt,

um die Überschneidungen detailliert zu beschreiben. Daraus wird das folgende Modell als Taxonomie von Problemen und Problemursachen im Physical Computing (Abbildung 4.1) abgeleitet. Die enthaltenen mehrdeutigen Problemursachen werden auf dieser Grundlage durch Beispiele untermauert.

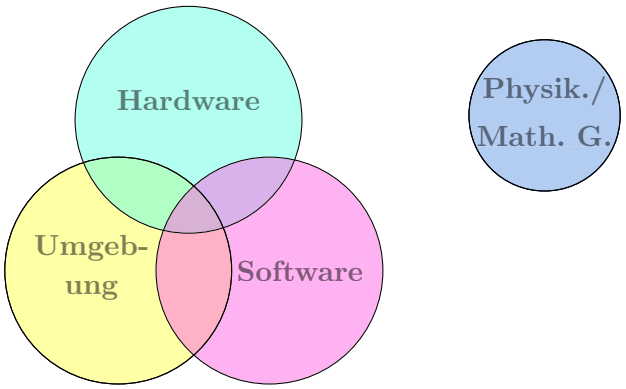


Abbildung 4.1: Empirisches Modell der auftretenden Problemursachen während der Arbeit mit Physical-Computing-Geräten nach Schulz und Pinkwart (2017)

Kategorie Hardware/Software Diese Kategorie kann durch die Überschneidung beider zuvor beschriebenen Kategorien (Hardware, Software) definiert werden. Das vorliegende Problem wird erzeugt, da die Systemkomponenten Hardware und Software nicht aufeinander abgestimmt sind. Die Anpassung in einem der Bereiche kann zur Lösung des Problems führen. Die aufgetretenen Unterkategorien, sind wie folgt definiert:

5a In dem geschriebenen Programm (Software) wurden die Spezifikationen der Hardware nicht berücksichtigt (z. B. Messbereiche von Sensoren).

5b Der in dem Programm festgelegte Schwellwert ist für die Konstruktion des Geräts nicht geeignet (z. B. wurde die eigene Länge des Roboter nicht berücksichtigt).

5c Die Software spricht Bauteile des Roboters an (Sensoren, Aktoren), die nicht/fehlerhaft funktionieren oder nicht angesprochen werden können (z. B. Kabel für einen Sensor wurden an den falschen Port angeschlossen oder an einem anderen Port als im Programm definiert).

5d Die Konstruktion des Geräts ist ungeeignet, so dass Programmteile negativ beeinflusst werden und das Programm nicht mehr den angestrebten Output hervorruft (z. B. Roboter fährt über seine eigenen Kabel und seine Bewegungsrichtung wird verändert).

Beispiel 4.3.1 *Kategorie Hardware/Software:*
Abbildung 4.2 skizziert ein mehrdeutiges Problem (das mehrere Problemursachen hat),

welches in den Studien aufgetreten ist. In diesem Szenario sollte ein LEGO Mindstorms-Roboter (in der Standardbauweise, vgl. Abbildung 3.2) um eine Kiste herum fahren und dafür ausschließlich den Ultraschallsensor nutzen. Der Sensor ist nach vorne in Fahrtrichtung ausgerichtet und der Messbereich hat einen Öffnungswinkel von maximal 20 Grad horizontal (vgl. graue Strahlen in Abbildung 4.2). In der Standardbauweise sind ebenfalls zwei Berührungssensoren vorne an dem Roboter angebracht und beide ragen leicht über die Außenkante des Roboters hervor. In einem geeigneten Programm sollte der Roboter rückwärts zurück fahren, sobald die Kiste vor dem Roboter erkannt wurde. Bei der hier aufgezeigten Position des Roboters zur Kiste ist es jedoch nicht möglich, die Kiste mittels Ultraschallsensor zu erkennen und deshalb fährt der Roboter gegen die Kiste. In diesem Fall wurden zumeist die Messgrenzen des Sensors vernachlässigt und die SuS nahmen an, dass ihr Programm für die Konstruktion des Roboters geeignet wäre. Daraus ergibt sich für die Lernenden die Entscheidung, ob sie in der Komponente Software oder Hardware Änderungen vornehmen. Diese könnten einerseits eine Veränderung des Programms sein (Software), in der der Roboter einen größeren Abstand zur Kiste hält und dadurch nicht in diese Position gerät. Andererseits wird die Hardware verändert und die Berührungssensoren werden entfernt. Dadurch verringert sich die Wahrscheinlichkeit an der Kiste hängen zu bleiben.

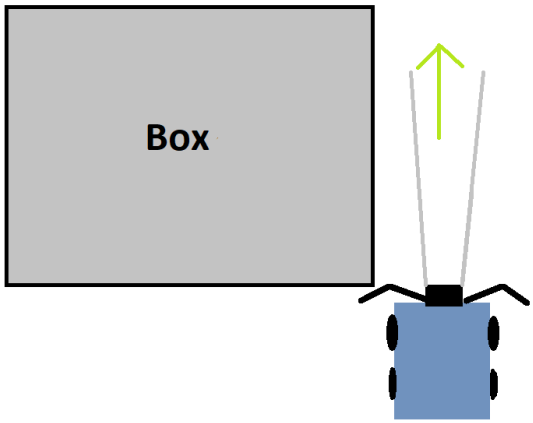


Abbildung 4.2: Darstellung eines mehrdeutigen Problems im Bereich Hardware/Software

Kategorie Hardware/Umgebung Diese Kategorie zeichnet sich durch die Überschneidung der Kategorien Hardware und Umgebung aus. Ähnlich wie bei den zuvor beschriebenen sich überschneidenden Kategorien, führt die ungenügende Berücksichtigung von Anforderungen beider Kategorien zu einem Problem. Veränderungen in dem Bereich Hardware oder Umgebung können das Problem lösen. Als Unterkategorien wurden die Folgenden identifiziert:

6a (Natürliche und menschliche) Einflüsse der Umwelt wirken sich auf die Funktion von Sensoren aus (vgl. Beispiel Hardware/Umgebung).

6b Eingreifen des Menschen verursacht Fehlfunktion/Defekt von Bauteilen des Geräts (z. B. Servomotor defekt durch das manuelle Drehen an den Rotorblättern).

6c Spezifikationen der Hardware werden bei Veränderungen der Umwelt außer Acht gelassen (z. B. Startposition des Roboters wird für fälschlich angenommenen Messwerte des Ultraschallsensors gewählt).

Beispiel 4.3.2 *Kategorie Hardware/Umgebung:*
Die SuS konstruierten LED-Lampen und einen Lichtsensor nebeneinander auf einem Steckbrett (vgl. Abbildung 4.3). Wenn sie den Lichtsensor mit einer Taschenlampe anstrahlten, sollte eine grüne LED-Lampe aufleuchten. Einigen SuS fiel es schwer zu erkennen, ob die grüne Lampe leuchtet, da sie diese ebenfalls mit der Taschenlampe anstrahlten. Teilweise bewerteten sie ihre Lösung als fehlerhaft, da vermeintlich nicht der erzielte Output eintraf. Es ist davon auszugehen, dass die SuS nicht wussten, worin der Fehler lag. An dieser Stelle war es möglich, Veränderungen in der Konstruktion oder der Umgebung vorzunehmen. Auf dem Steckbrett könnten die LED-Lampen und der Lichtsensor mit einer größeren Entfernung zueinander angebracht werden. Es wäre ebenfalls möglich, einen geringeren Widerstand für die LED-Lampen zu wählen, damit sie heller leuchten. Die Veränderung der Umgebung würde in diesem Fall bedeuten, dass die SuS darauf achten lediglich den Sensor zu beleuchten und die Taschenlampe dementsprechend halten. Das heißt, der menschliche Einfluss auf die Umgebung wird verändert.

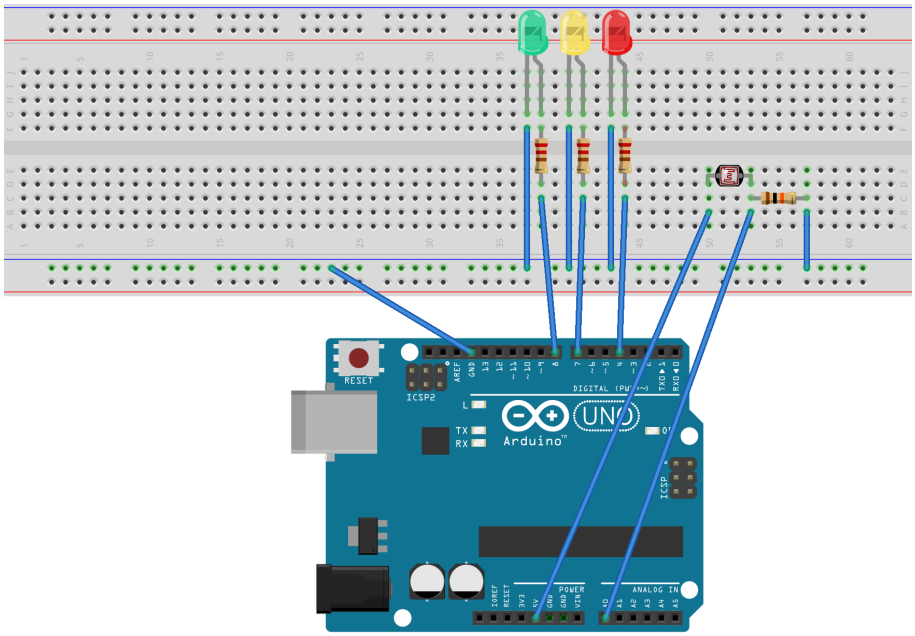


Abbildung 4.3: Darstellung eines mehrdeutigen Problems im Bereich Hardware/Umgebung

Kategorie Software/Umgebung Diese Kategorie beschreibt die Überschneidung der Bereiche Software und Umgebung. Es kann nicht auf eine der Kategorien eingegrenzt

werden, da die Programmierung der Software ein spezifisches Verhalten der Umgebung erfordert und umgekehrt. Die folgenden Unterkategorien wurden in den Studien ermittelt:

7a Der Schwellwert in der Software ist nicht an die Umweltbedingungen angepasst (z. B. ist der Schwellwert für den Lichtsensor so ungünstig gewählt, dass die LED-Lampe flackert, anstatt zu leuchten).

7b Der Einfluss des Menschen in die Umwelt wurde in der Software nicht berücksichtigt (z. B. ständige Veränderung der Startposition des Roboters durch den Menschen).

Beispiel 4.3.3 *Kategorie Software/Umgebung:*
Die SuS schrieben ein Programm, bei dem je nach Helligkeit verschiedene LED-Lampen aufleuchten. Die Helligkeit wurde mit Hilfe eines Lichtsensors gemessen. Es handelt sich damit um das gleiche Szenario wie in dem zuvor beschriebenen Beispiel, jedoch wird ein anderes Problem fokussiert. Um die Aufgabe zu lösen, müssen verschiedene Grenzwerte für die Stufen der Lichthelligkeit ermittelt und in der Software gesetzt werden. Einige SuS hatten das Problem, dass sich die gemessenen Helligkeitswerte (mit dem sie die Grenzen festlegten) bei dem Test des Programms als ungeeignet herausstellten. Das hing damit zusammen, dass sie bei dem Testfall „Sensor abgedeckt“ den Sensor nicht vollständig verdeckten (versucht mit einem Finger). In einem anderen Fall lehnten sie sich bei einigen Messungen über den Sensor, so dass das Umgebungslicht verändert wurde. Hieraus ergibt sich als erste Lösungsmöglichkeit, dass die SuS darauf achten, das Umgebungslicht möglichst konstant zu halten und den Sensor vollständig abzudecken. Auf Seiten der Software könnten jedoch ebenfalls die Grenzwerte angepasst werden, z. B. können sie großzügiger oder als Intervall gesetzt werden. Eine weitere Lösung auf der Softwareebene ist das Addieren von mehreren Werten und die Berechnung eines Mittelwerts, damit das Ergebnis weniger anfällig für Schwankungen der Lichtverhältnisse ist.

Kategorie System In der Kategorie System sind alle Komponenten des Systems (Software, Hardware und Umgebung) Teil der Problemlösung. Das Problem ist so vielfältig, dass eine Veränderung in einer der Komponenten (in einem angemessenen Aufwand) zur Lösung führen kann.

8a Der Algorithmus ist ungeeignet für die Konstruktion des Geräts und die durch den Menschen verursachte Veränderung der Umgebung (vgl. Beispiel I).

8b Der Algorithmus ist ungeeignet für die Konstruktion des Geräts und die vorherrschenden Bedingungen der Umgebung (vgl. Beispiel II).

Beispiel 4.3.4 *Kategorie System, Beispiel I:*
Ein Beispiel, das alle Systemkomponenten betrifft, ist die Positionierung des Ultraschallsensors. In der Standardbauweise der LEGO Mindstorms-Roboter ist dieser erhöht und schräg hinter dem Brick angebracht. Ein häufiges Problem zeigt sich bei dem Umfahren

der Kiste darin, dass die meisten SuS den Roboter zunächst gerade auf die Kiste zufahren lassen wollten und sich dieser zu einer Seite drehen sollte, sobald ein Hindernis in circa 20 cm Entfernung auftritt. Viele SuS beachteten jedoch nicht, dass sie ihre Hand beim Starten des Programms (auf dem Brick) vor dem Ultraschallsensor hatten. Deshalb drehte sich der Roboter teilweise direkt zur Seite und fuhr dann geradeaus, was sich viele SuS nicht erklären konnten (vgl. Abbildung 4.4). Dieses Problem liegt zugleich in den drei Hauptkategorien Software, Hardware und Umgebung. Auf der Softwareebene könnte das Programm verändert werden, so dass der Roboter zunächst geradeaus fährt oder wartet und erst im Anschluss den Sensorwert des Ultraschallsensors abfragt. Die Position des Sensors (Hardware) könnte ebenfalls verändert werden, so dass die SuS nicht mehr darauf achten müssen, wie sie den Startknopf auf dem Brick betätigen ohne dabei die Hand vor dem Sensor zu haben. Dem schließt sich die Problemursache der Umgebungseinflüsse durch den Menschen an. Würden die SuS darauf achten den Startknopf von der Seite aus und nicht von oben zu betätigen, so würden sie nicht in das System eingreifen.

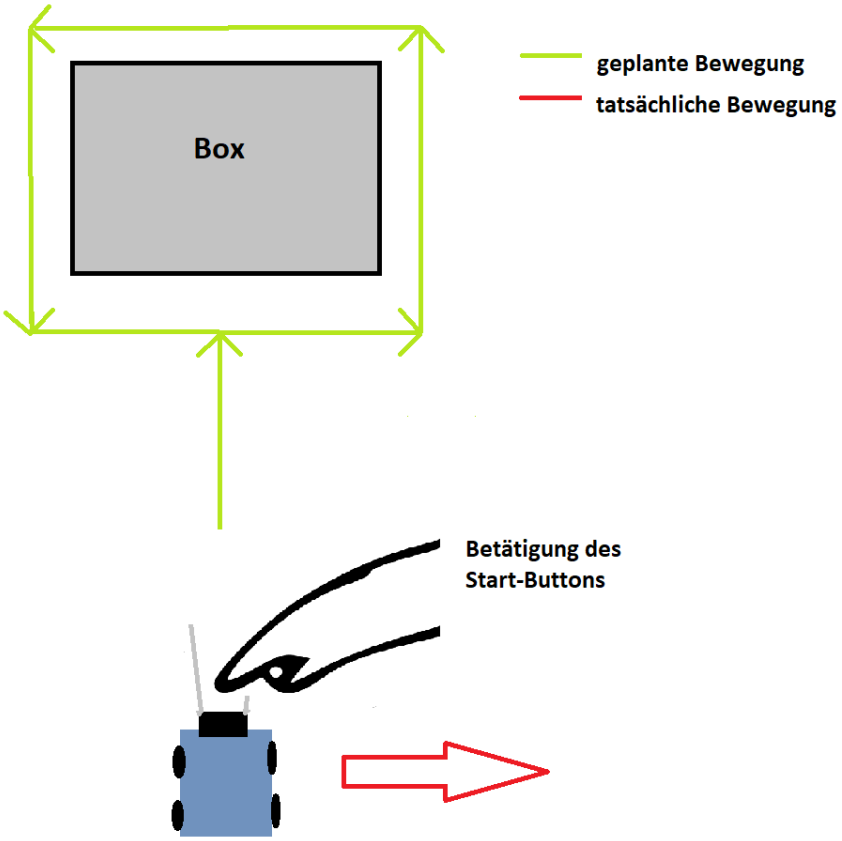


Abbildung 4.4: Darstellung eines mehrdeutigen Problems im Bereich System

Beispiel 4.3.5 *Kategorie System, Beispiel II:*
Versuchten die SuS eine exakte 90-Grad-Drehung zu implementieren, zeigten sich eben-

falls Probleme, die sich auf das gesamte System beziehen. Einerseits kamen Umweltfaktoren zum Tragen, z. B. ein rutschiger oder staubiger Boden, was das Fahrverhalten des Roboters beeinträchtigte. Hardwareseitige Einflüsse sind, dass die Motoren leichte Abweichungen in der Umsetzung von elektrischer Energie in mechanische Energie aufweisen. Des Weiteren können die Radprofile unterschiedlich abgenutzt sein. Bei der Implementation der Software muss darauf geachtet werden, dass die Gradzahlen der Umdrehung für eine spezielle Radgröße angegeben sind. Sobald davon abgewichen wird, muss die Gradzahl dementsprechend im Programm berechnet werden.

Ob ein Problem in einer der Problemursachen oder in mehreren gelöst werden kann, hängt maßgeblich von dem vorliegenden Problem ab. Beispielsweise können durch Anpassungen in der Software vorhandene Umgebungseinflüsse reduziert werden. Prinzipiell scheinen solche Probleme für SuS besonders schwer lösbar zu sein. Das liegt einerseits darin begründet, dass die Problemursachen identifiziert werden müssen und andererseits darin, dass entschieden werden muss, in welcher Problemursache zuerst Adaptionen vorgenommen werden. Hierbei sind manche Problemursachen leichter zu beheben als andere.
Die hier vorgenommene Einordnung von vorliegenden Problemen und ihren zugehörigen Problemursachen basiert auf den Beobachtungen von SuS. Die Kategorisierung eines Problems zeigt die beteiligten Problemursachen auf. Dadurch werden auch die Kategorien verdeutlicht, in denen Änderungen vorgenommen werden können, um das Problem zu lösen. Bei der Untersuchung der Problemursachen wurde sich auf die beschränkt, die innerhalb der Studien sichtbar und die teilweise von den SuS als Ausgangspunkt für die Lösung von Problemen genutzt wurden. Viele Probleme lassen sich der Kategorie System zuordnen, doch die Unterscheidung ist hier sinnvoll, da sie den SuS eine systematische Reflexion für mögliche Lösungsansätze verschafft.

4.3.3 Quantitative Beschreibung der Problemursachen

Die aufgetretenen Probleme wurden bereits qualitativ beschrieben. In diesem Abschnitt wird nun die quantitative Verteilung der zugrundeliegenden Problemursachen aufgezeigt. Diese Werte geben Hinweise auf die Verteilung von Problemen auf die Problemursachen, deren Quantifizierung jedoch nicht von einem zweiten Codierer gesichert worden ist. Aufgrund der geringen Stichprobe genügt die Darstellung nicht den statistischen Gütekriterien, zeigt jedoch Tendenzen auf, die für weitere Forschung in diesem Bereich genutzt werden können.
Die Tabelle 4.2 enthält die Gesamtverteilung aller gefundenen Probleme und deren Zuordnung zu den Problemursachen. In Abbildung 4.5 werden die in Tabelle 4.2 abgebildeten Werte visualisiert. Die genutzte Farbcodierung der Problemursachen entspricht dabei den Farben der vorgestellten Problemtaxonomie (vgl. Abbildung 4.1; Cyan $\widehat{=}$ Hardware, Gelb $\widehat{=}$ Umgebung, Magenta $\widehat{=}$ Software, Orange $\widehat{=}$ Software/Umgebung, Violett $\widehat{=}$ Hardware/Software, Grün $\widehat{=}$ Hardware/Umgebung, Braun $\widehat{=}$ System).
In den folgenden Abbildungen (4.5, 4.6 und 4.7) ist die Fläche der grafischen Objekte proportional zur Anzahl der adressierten Problemursachen in der jeweiligen Kategorie.

Tabelle 4.2: Summe aller aufgetretener Probleme und Zuordnung zu den Problemursachen

	Software	**Hardware**	**Umgebung**	**System**
Software	45	30	11	
Hardware		63	4	
Umgebung			7	
System				8

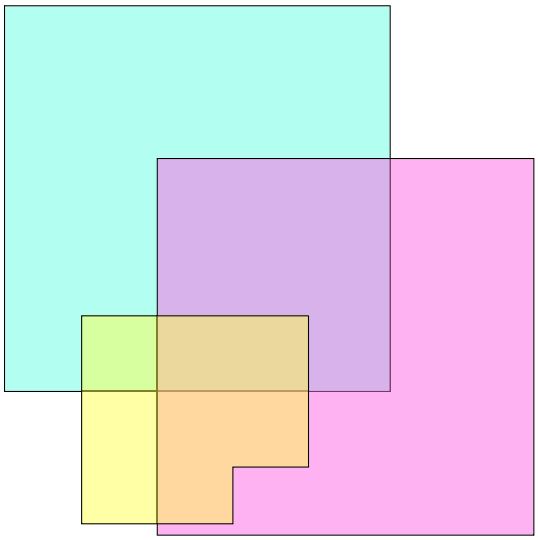


Abbildung 4.5: Visualisierung aller aufgetretener Probleme mit Zuordnung zu den Problemursachen

Insgesamt wurde bei den Studien $1 - 4$ eine Anzahl von 168 Problemen identifiziert. Viele dieser Probleme lagen in den Hauptproblemursachen Software und Hardware (vgl. $N = 45$ und $N = 63$). An dritter Stelle der häufigsten Problemursachen befindet sich die Überschneidung der Problemursachen Software und Hardware. Die übrigen Problemursachen (Software/Umgebung, Hardware/Umgebung, Umgebung und System) fallen in ihrer Häufigkeit wesentlich geringer aus und lassen in der Ansicht über alle Geräte hinweg keine weiteren Schlüsse zu.

Aus diesem Grund wird nun die Verteilung der Probleme bei der Arbeit mit LEGO Mindstorms-Robotern numerisch (vgl. Tabelle 4.3) und illustriert (vgl. Abbildung 4.6) dargestellt. Die Kategorie mathematische/physikalische Grundlagen wurde in der Tabelle nicht dargestellt, jedoch in der Datenanalyse codiert. Dieser Kategorie zuzuordnen sind zehn aufgetretene Probleme. Bei der Arbeit mit LEGO Mindstorms-Robotern konzentrieren sich die Probleme auf den Bereich Software ($N = 30$) und es gibt ebenfalls viele Probleme im Bereich Software/Hardware ($N = 15$). Hingegen sind relativ wenig Probleme in der Hardware ohne Überschneidungen zu anderen Problemursachen zu verorten ($N = 5$). Umgebungsprobleme und der Bereich Hardware/Umgebung traten selten auf, allerdings gab es vergleichsweise viele Probleme im Bereich Software/Umgebung ($N = 10$). Probleme, die dem gesamten System zuzuordnen sind, traten siebenmal auf.

Tabelle 4.3: Summe aufgetretener Probleme bei der Arbeit mit LEGO Mindstorms-Robotern und Zuordnung zu den Problemursachen

	Software	**Hardware**	**Umgebung**	**System**
Software	30	15	10	
Hardware		5	2	
Umgebung			2	
System				7

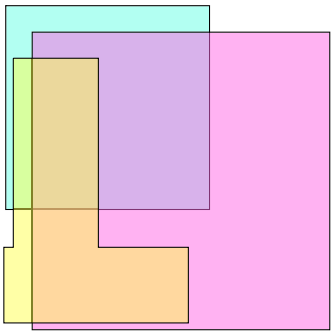


Abbildung 4.6: Visualisierung von bei der Nutzung von LEGO-Robotern aufgetretenen Problemen mit Zuordnung zu den Problemursachen

Basierend auf dieser Verteilung kann die folgende Hypothese aufgestellt werden: *bei der Arbeit mit modularen Physical-Computing-Geräten, wie z. B. LEGO Mindstorms-Robotern, treten weniger ausschließlich hardwareseitige Probleme auf.*

Bei der Betrachtung der aufgetretenen Probleme bei der Arbeit mit Arduino-Mikrocontrollern (vgl. Tabelle 4.4 und Abbildung 4.7) ist der Bereich Hardware die hauptsächliche Problemursache mit $N = 58$. Diesem Bereich folgen die Kategorien Software und Software/Hardware mit jeweils 15 aufgetretenen Problemen. Die Kategorie Umgebung trat relativ selten ($N = 5$) ohne eine Überschneidung zu anderen Problemursachen auf, wobei die Kategorien Umgebung/Hardware zweimal und Umgebung/Software einmal auftraten. Lediglich einmal ließ sich ein Problem allen Problemursachen zuordnen. Es wurde ein Problem codiert, das der Problemursache der mathematischen/physikalischen Grundlagen zuzuordnen ist. Bei dem Vergleich der LEGO-Roboter mit den Arduino-

Tabelle 4.4: Summe aufgetretener Probleme bei der Arbeit mit Arduino-Mikrocontrollern und Zuordnung zu den Problemursachen

	Software	**Hardware**	**Umgebung**	**System**
Software	15	15	1	
Hardware		58	2	
Umgebung			5	
System				1

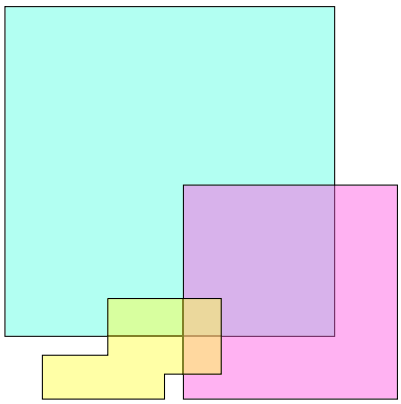


Abbildung 4.7: Visualisierung von bei der Nutzung von Arduino-Mikrocontrollern aufgetretenen Problemen mit Zuordnung zu den Problemursachen

Mikrocontrollern fällt auf, dass die häufigsten Probleme in den Hauptproblemursachen Software und Hardware liegen, gefolgt von Überschneidungen von Software/Hardware. Dabei sind bei der Arbeit mit LEGO Mindstorms-Robotern vorrangig Softwareprobleme aufgetreten, bzw. Überschneidungen der Software mit weiteren Problemursachen. Die SuS, die mit Arduino-Mikrocontrollern arbeiteten, hatten weniger Probleme in den mehrdeutigen Kategorien. Wird davon ausgegangen, dass das Lösen von Problemen, die mehrdeutige Problemursachen haben, schwieriger ist, so kommt man zu der folgenden Hypothese: *die*

Verwendung von Mikrocontrollern im Physical Computing erleichtert die Zuteilung von Problemen zu den Bestandteilen des Physical-Computing-Systems.

Es ist zu berücksichtigen, dass diese Roboter modular aufgebaut sind und hardwareseitig wenige bis keine Veränderungen vorgenommen werden müssen. Hingegen wurden bei den Arduino-Mikrocontrollern viele Hardwareprobleme oder Überschneidungen von Kategorien mit der Problemursache Hardware beobachtet. Die SuS hatten diverse Konstruktionsarbeiten vornehmen müssen, was jedoch den Programmieraufwand nicht reduzierte. Daher ist der starke Rückgang der Softwareprobleme überraschend. Probleme, die ausschließlich dem Bereich Hardware und der Umgebung zuzuordnen sind, wurden bei der Verwendung von Arduino-Mikrocontrollern öfter beobachtet als bei LEGO-Robotern. Von den drei Unterkategorien der Software (Programmcode, Programmierumgebung und Firmware) ist die Firmware auf dem Gerät kein Bereich mit dem die SuS direkt interagierten, da das Programm unverzüglich übertragen und ausgeführt wird, ohne zuvor Änderungen am Gerät vorzunehmen. Diese Unterkategorie spielte jedoch auch in den Studien mit LEGO-Robotern keine große Rolle. Daher ist davon auszugehen, dass ein Einfluss, der die direkte Interaktion mit der Firmware auf die Anzahl von Softwareproblemen beinhaltet, vernachlässigt werden kann. Eine mögliche Erklärung ist, dass der Unterschied der Altersgruppe und des Vorwissens in den Studien eine große Rolle spielte. Es arbeiteten SuS der Sekundarstufe I mit den Robotern, wobei die Gruppen mit Mikrocontrollern bereits die Sekundarstufe II besuchten. Ebenfalls handelte es sich in der Arduino-Studie um besonders interessierte und begabte SuS im Bereich der Physik. Daher konnten bei diesen kaum grundlegende Probleme bei der Konstruktion von Stromkreisen festgestellt werden. Möglicherweise hatten sie aufgrund ihrer Erfahrungen im naturwissenschaftlichen Problemlösen weniger Probleme mit dem logischen Denken beziehungsweise mit der Antizipation der Auswirkungen ihres Programms. Es ist davon auszugehen, dass die auftretenden Probleme stark mit der konkreten Aufgabe in Zusammenhang stehen. Jedoch sind die gestellten Aufgaben in ihrer Komplexität schwer zu vergleichen, da mit unterschiedlichen Geräten gearbeitet wurde. Hinzu kommt, dass der Prozess der Aufgabenbewältigung leichte Unterschiede unter den Geräten aufweist. Die Arduino-Mikrocontroller sind direkt an einem Computer angeschlossen und das Programm wird mit einem Knopfdruck übertragen und ausgeführt. Aus diesem Grund sind kleine Veränderungen wie z. B. das Setzen eines Grenzwerts unkompliziert und schnell realisiert. Es führte seltener zu einem Problem und wurde nicht als solches diskutiert.

4.4 Diskussion und Zusammenfassung

Ausgangspunkt für diese Untersuchung war die Annahme, dass ein Zusammenbringen von einem Computersystem mit der realen Welt mehr Problemursachen aufwirft als ausschließlich die Summe aus beiden Bereichen. Eine weitere aus der Literatur (vgl. Kafai et al., 2014b; Okita, 2014) abgeleitete Annahme ist, dass SuS insbesondere Probleme beim Lösen

von Physical-Computing-Aufgaben haben. Das kann auf die Vielfalt der Problemursachen zurückzuführen sein, deren Abgrenzung schwierig sein kann. Als Problemursachen wurden die Systemkomponenten des Physical Computing Software, Hardware und Umgebung identifiziert. Die Kategorie mathematische/physikalische Grundlagen wurde als separate Problemursache beschrieben und wurde in der Physical-Computing-Literatur bislang vernachlässigt. Im Rahmen dieser Arbeit wurde die Kategorie aufgegriffen und als Problemursache beschrieben, da sie in zukünftigen Untersuchungen Berücksichtigung finden sollte und ein Bestandteil der Arbeit mit Physical-Computing-Geräten ist. Der Fokus blieb jedoch auf den drei Systemkomponenten des Physical-Computing-Systems. Darüber hinaus wurden weitere Problemursachen in Form von Überschneidungen der Hauptproblemursachen (mehrdeutige Problemursachen) entdeckt. Es gibt Hinweise darauf, dass das Auftreten von Problemen mit verschiedenen Problemursachen im Zusammenhang mit den Geräteeigenschaften steht. So mussten die SuS, die mit Arduino-Mikrocontrollern arbeiteten, verschiedene Konstruktionen von Schaltkreisen vornehmen. Daher hatten sie mehr Probleme in diesem Bereich als SuS, die wenige Konstruktionen vornehmen mussten, wie z. B bei der Arbeit mit LEGO-Robotern.
Ausgehend von den untersuchten Kategorien und speziellen Problemen können Implikationen für die Schulpraxis und die Forschung abgeleitet werden. In der Schule könnten die SuS davon profitieren, wenn ihnen die Problemursachen und Systemkomponenten zuvor vorgestellt werden (vgl. der Cognitive-Load-Theory nach Sweller et al., 1998). Ein Raster zur Einordnung der auftretenden Probleme zu Problemursachen für die SuS ist ebenfalls möglich. Es kann dazu dienen, die auftretenden Probleme zu kategorisieren und ein Verständnis für Ursachen des Problems und das Physical-Computing-System zu entwickeln. Des Weiteren kann es das Ziehen von Analogien zu ähnlichen Problemen erleichtern, sofern diese Probleme zuvor intensiv analysiert und diskutiert wurden. Die Ergebnisse deuten darauf hin, dass abhängig von den Bestandteilen und dem Aufbau des gewählten Physical-Computing-Geräts sowie den damit verbundenen Aufgabenbereichen, die Problemursachen in der Software und Hardware angesiedelt sind. Diese beiden Kategorien können genügen, um eine Vielzahl von Problemen zu beschreiben. Deshalb kann die Unterscheidung von diesen Kategorien als Einstieg zur Identifikation von Problemursachen genutzt werden, wovon ausgehend weitere Dimensionen hinzugenommen werden können.
Für die Konstruktion von Physical-Computing-Geräten (bzw. Construktions Kits) und dazugehöriger Aufgaben, sollten die Problemursachen berücksichtigt werden. Aufgetretene Probleme bei der Konstruktion von Stromkreisen oder der Auswahl von (Input-/Output-)Ports können methodisch unterstützt werden. Möglichkeiten dafür sind Beschriftungen durch Schablonen, die auf den Mikrocontroller aufgelegt werden und Anschlüsse detaillierter beschreiben. Die Entwicklung von intuitivem Design des Mikrocontroller-Boards und dem zugehörigen Material ist ein weiterer Ansatz. In der Informatikdidaktik sollten prinzipielle Überlegungen zur Reduktion der kognitiven Beanspruchung im Physical Computing zum Gegenstand der Forschung werden. Bislang ist offen,

welche mathematischen und physikalischen Fähigkeiten und Fertigkeiten SuS benötigen, um Physical-Computing-Aufgaben zu bewältigen. Es wurde der Bedarf aufgezeigt, dass mathematische/physikalische Grundlagen für die Arbeit mit Physical-Computing-Geräten untersucht werden sollten, da SuS Probleme in dieser Kategorie hatten. Direkte Voraussetzungen für die erfolgreiche Bearbeitung von Physical-Computing-Aufgaben sind die Ermittlung von Messgrenzen eines Sensors und die Verwendung von Operatoren. Dabei erscheint die Verbindung mit Forschungsergebnissen der wissenschaftlichen Erkenntnisgewinnung aufgrund der inhaltlichen und prozessualen Gemeinsamkeiten besonders geeignet zu sein.

Weiterführende Forschung im Bereich des Physical Computing sollte sich mit der Fragestellung beschäftigen, wie die SuS in diesem Problemlöseprozess unterstützt werden können. Einen ersten Ansatz dafür untersucht die 3. Forschungsfrage der hier vorliegenden Arbeit: *Welche Arten von Hilfestellungen sind effektiv für Schülerinnen und Schüler, um den Physical-Computing-Prozess zu unterstützen?* (Kapitel 5). Die Weiterentwicklung dieses Forschungsfeldes kann dazu führen, dass in Abhängigkeit von Geräteeigenschaften, Lerngruppen und Aufgabenstellungen gezielt Problemursachen herausgearbeitet werden, die einen besonders großen Einfluss auf die Physical-Computing-Aktivitäten haben. Anschließend kann für Lehrende und Lernende das Augenmerk auf diese Bereiche gelegt und gezielt durch didaktische und methodische Hilfestellungen unterstützt werden. Ähnlich wie in der naturwissenschafts-didaktischen Forschung der wissenschaftlichen Erkenntnisgewinnung erscheint es vielversprechend, die auftretenden Probleme den Phasen des Physical-Computing-Prozesses zuzuordnen (vgl. Masnick und Klahr, 2003), um in der Unterrichtspraxis entsprechende Hilfestellungen bereitstellen zu können. Dafür sind jedoch diverse weitere Studien notwendig, um möglichst allgemeingültige Aussagen darüber treffen zu können.

Es ist darauf zu achten, dass die hier aufgeführten quantitativen Ergebnisse nicht auf die verwendeten Gerätetypen generalisierbar sind. Durch die Analyse der quantitativen Verteilung von Problemursachen in Kapitel 4.3.2 wurde deutlich, dass auftretende Probleme in Abhängigkeit zu den folgenden Faktoren stehen können: erteilte Aufgabenstellung, Physical-Computing-Geräten, Programmierumgebungen und den Voraussetzungen der SuS. Diese Faktoren wurden in dieser Arbeit variiert, jedoch zu diesem Zeitpunkt nicht systematisch für die Faktoren evaluiert. Es wurden jedoch alle aufgezeigten Problemursachen unabhängig von den verwendeten Geräten wiedergefunden. Zur Validierung der quantitativen Verteilung von Problemursachen sind weitere Studien notwendig, die die zuvor angesprochenen Variablen variieren. Darüber hinaus sollten weitere Merkmale in der Codierung herangezogen werden, um eine genaue Unterscheidung vornehmen zu können, auf welche Kompetenzbereiche die Probleme zurückzuführen sind.

Kapitel 5

Hilfestellungen für den Physical-Computing-Prozess

Nachdem konkrete Probleme und deren Ursachen in Kapitel 4 identifiziert wurden, sollen in diesem Kapitel konkrete Unterstützungsmöglichkeiten für den Physical-Computing-Prozess untersucht werden. Das wird durch die 3. Forschungsfrage dieser Arbeit realisiert, die wie folgt lautet:

Welche Arten von Hilfestellungen sind effektiv für Schülerinnen und Schüler, um den Physical-Computing-Prozess zu unterstützen?

Das Ziel ist, Hilfestellungen für Schülerinnen und Schüler (SuS) bei Physical-Computing-Aktivitäten zu entwickeln und zu evaluieren. Ähnlich wie in der Forschung zur wissenschaftlichen Erkenntnisgewinnung konnten auch im Physical-Computing-Prozess diverse Probleme von SuS und Problemursachen identifiziert werden (vgl. Kapitel 4). Das Potential von Physical-Computing-Geräten im Unterricht wird als gewinnbringend eingestuft. Innerhalb der Physical-Computing-Forschung existieren bislang jedoch kaum Ansätze für Unterstützungsmöglichkeiten oder eine empirische Evaluation von Unterstützungen. Wie auch im Bereich der Unterstützungstools für die wissenschaftliche Erkenntnisgewinnung (vgl. Kapitel 2.2.2) kann in der Informatik davon ausgegangen werden, dass die Unterstützung des Problemlöseprozesses einen positiven Einfluss auf die Erkenntnisgewinnung hat. Erkenntnisse in der Informatik umfassen zum Beispiel Prinzipien, wie das EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe) oder auch prozessuale Kenntnisse, wie die zur Softwareentwicklung. Forschungsergebnisse zeigen, dass die extrinsische Motivation der SuS insbesondere zum Beginn der Arbeit mit Robotern sehr hoch ist und anschließend zurückgeht. Zum Teil wird diese Beobachtung als Zeichen des Anstiegs der intrinsischen Motivation bewertet (vgl. McWhorter und O'Connor, 2009). Jedoch kann die Bearbeitung von schwierigen Aufgaben ebenfalls demotivierend sein und es wurde keine Evidenz dafür gefunden, wie lange die Motivation der SuS im Umgang mit Physical-Computing-Geräten auf einem hohen Level gehalten werden kann. Um zusätzlich die Motivation zu unterstützen

und somit eine positive Wirkung bei der Bewältigung von Physical-Computing-Aufgaben zu erzielen, kann das adaptive Feedback von kognitiven Tutorensystemen eingesetzt werden (vgl. McLaren et al., 2008). Ein Tutorensystem kann den Unterricht auch insofern unterstützen, dass die Lehrkraft mehr Zeit für individuelle Förderung hat und der Frontalunterricht zeitlich reduziert werden kann (vgl. Koedinger et al., 2006).
Von Anderson et al. (1995, S. 202) werden Tutorensysteme wie folgt beschrieben: „We now conceive of a tutor as a learning environment in which helpful information can be provided and useful problems can be selected. We are able to take actions that facilitate learning because we possess a cognitive model of where the student is in that task.“ Es existieren verschiedene Ausprägungen von Tutorsystemen, z. B. *intelligente Tutorensysteme (ITS)*, denen wiederum *kognitive Tutorsysteme* zugeordnet werden (Koedinger und Aleven, 2007). Letztere setzen sich zum Ziel, Lernenden ein gezieltes und individuelles Feedback zu geben, das vergleichbar mit einer privaten Lehrkraft ist. Bei den Systemen handelt es sich jedoch um ein computergestütztes System, welches unabhängig von Klassenräumen eingesetzt werden kann. Koedinger und Aleven (2007) erklären, dass die kognitiven Tutoren der kognitionspsychologischen Theorie des Problemlösens und Lernens folgen. Ein weiteres Merkmal ist, dass die Lernenden schrittweise geleitet werden und ein regelmäßiges Feedback erhalten.
Die bereits angesprochenen intelligenten Tutorensysteme könnten geeignet sein, um Lernende innerhalb des Physical-Computing-Prozesses zu unterstützen.
Aus diesem Grund wird in Kapitel 5.1 zunächst der Forschungsstand von Unterstützungsmöglichkeiten in Form von Tutorensystemen dargelegt. Diese Literaturrecherche bildet die Phase Analysis & Exploration III und legt die Grundlage, um Unterstützungstools für Physical Computing zu entwickeln. Die konkrete Planung der Untersuchung sowie die Konstruktion von Untersuchungsdesigns wird in Kapitel 5.2 vorgenommen (Design & Construction III). In den Kapiteln 5.3 und 5.4 werden drei Studien vorgestellt, die zum Ziel haben, zwei (in dieser Arbeit konzipierte) Hilfestellungen zu evaluieren (Evaluation & Reflection III). Diese sollen SuS bei der Arbeit mit Physical-Computing-Geräten unterstützen. Ein Überblick über die Ziele der einzelnen Studien dieses Kapitels wird in Tabelle 5.1 gegeben.

Tabelle 5.1: Überblick von Studien zur Beantwortung der 3. Forschungsfrage

Studienname	**Intensivierung der Evaluationsphase**	**Wizard-of-Oz I**	**Wizard-of-Oz II**
Kapitel	5.3	5.4.1	5.4.6
Studiennummer	3	5	6
PhC-Geräte	LEGO-Roboter	LEGO-Roboter, Nao-Roboter	LEGO-Roboter, Nao-Roboter
Wizard	-	sichtbar	verdeckt
Hilfestellung	*Evaluationsphase digital*	*Problemtaxonomie gestuft*	*Problemtaxonomie gestuft*
Art der Unterstützung	prozessual	direkte Instruktion	direkte Instruktion
Ziel	Intensivierung der Evaluationsphase zur Unterstützung des PhC-Prozesses	Direkte Instruktionen bzgl. aufgetretener Probleme zur Unterstützung des PhC-Prozesses	Anwendbarkeit der Hilfestellung als Grundlage eines kognitiven Tutorensystems testen

5.1 Forschungsergebnisse zur Gestaltung von Feedback durch Tutorensysteme

In Abschnitt 2.2.2 wurde bereits dargestellt, dass die Art des Feedbacks in der Forschung der wissenschaftlichen Erkenntnisgewinnung untersucht wurde. In diesem Abschnitt werden Forschungsergebnisse aufgezeigt, die Hinweise und Designvorschläge für die Konstruktion von Hilfestellungen in Form von intelligenten Tutorensystemen geben. Diese werden anschließend auf den Bereich des Physical Computing übertragen. Es existieren diverse Fragestellungen in diesem Forschungsfeld, wobei lediglich die für diese Dissertation relevanten Fragestellungen betrachtet werden.
Im Folgenden werden Ergebnisse bezüglich des Zeitpunktes und der Intensität des Feedbacks vorgestellt. Dabei handelt es sich vor allem um sogenannte *kognitive Tutoren*.

Wird das Feedback angefragt oder erteilt? Aleven et al. (2003) ziehen auf Basis eines Literaturreviews den Schluss, dass LernerInnen nicht wissen, wann sie Hilfe benötigen und diese suchen sollten. Dies ist unabhängig von spezifischen Lernumgebungen und damit auch zutreffend über Klassenraumsituationen hinaus. In vielen Studien stellte sich heraus, dass die SuS Hilfestellungen ineffektiv nutzten oder ignorierten. Eine Vielzahl von Studien mit SuS der Mittel- bis Oberschule zeigen auch, dass SuS Schwierigkeiten haben den richtigen Zeitpunkt zu finden, um Hilfe zu suchen (Aleven et al., 2006; Baker et al., 2004). Diese Ergebnisse motivieren dazu, das Design von Hilfestellungen intensiv zu untersuchen und bei der Entwicklung von kognitiven Tutoren zu hinterfragen. Eine wichtige Frage des Designs adressiert den Zeitpunkt des Feedbacks.

Wann wird das Feedback gegeben? Viele kognitive Tutorensysteme geben erst ein Feedback an die NutzerInnen, wenn diese explizit danach fragen. Koedinger und Aleven bezeichnen die Entscheidung, zu welchem Zeitpunkt Lernsysteme unterstützen sollten, als *Assistance Dilemma* (Koedinger und Aleven, 2007). Die Frequenz dieser Unterstützung spielt dabei ebenfalls eine Rolle bei der Entscheidung, ob die NutzerInnen lediglich Anregungen erhalten und die Möglichkeit bekommen die Lösung selbst zu entwickeln oder ein permanentes Feedback erhalten. In dieser Untersuchung wurde ein interaktives oder nicht-interaktives Feedback gegeben, wobei die NutzerInnen selbst Hinweise im System anfordern konnten. Diese wurden in fünf Stufen erteilt und immer präziser. Koedinger und Aleven kommen dabei zu dem Schluss, dass noch weitere Untersuchungen mit unterschiedlichen Bedingungen durchgeführt werden sollten, bis die Frage nach dem richtigen Maß an Unterstützung beantwortet werden kann.

Wie intensiv sollte unterstützt werden? Einen Ansatz zur Verbesserung des wissenschaftlichen Denkens wählten z. B. McLaren et al. (2014) mit der Entwicklung einer Lernumgebung, die evolutionäre Prozesse simuliert. Bei diesem Ansatz wurde untersucht,

in welcher Intensität Hilfestellungen vorliegen sollten, damit die SuS Aufgaben bewältigen können. Es wurde unterschieden in drei verschiedene Hilfestellungen:

1. keine,
2. Fehler aufzeigen, ein Textfeedback zu den Fehlern ausgeben, Hilfestellungen,
3. gleiche Hilfestellung wie bei 2, jedoch vorgeschaltet und somit präventiv.

Das Resultat der Pilotstudie war, dass die besser abschneidenden SuS mehr Hilfestellungen nutzten als die SuS, die die Hilfestellung des Programms dringender gebraucht hätten. Zur Beurteilung der SuS wurde bewertet, ob sie ein Level im Programm abgeschlossen haben und ob durchgeführte Schritte zum Erreichen eines Ziels führten oder nicht. Die Unterscheidung von gut und schlecht Abschneidenden wurde von der Lehrkraft vorgenommen. Diesbezüglich wurden Aspekte, wie z. B. Inhaltsverständnis und Fähigkeiten zur Erkenntnisgewinnung betrachtet. McLaren et al. äußern die Vermutung, dass dieser Effekt im Zusammenhang mit der aktuellen Version des genutzten Programms stehen kann. Diese unterstützt nur anfängliche Phasen des Lernprozesses, allerdings benötigen schlechter abschneidende SuS möglicherweise in späteren Phasen ebenfalls Unterstützung. Als weiterer Faktor wird genannt, dass bessere SuS genauer wissen, wann sie Unterstützung benötigen und über bessere metakognitive Fähigkeiten verfügen.

Die Analyse von Tutorensystemen durch Wizard-of-Oz-Studien. Wizard-of-Oz-Studien (vgl. Kapitel 5.2.1) können ein Mittel sein, um die Effektivität dieser Hilfestellungen zu ermitteln. Beispielsweise entwickelten McLaren et al. (2008) das Tool *VLab* zur Unterstützung von konzeptionellem Verständnis in der Chemie. Sie implementierten eine adaptive und nicht adaptive Version, wobei die Adaption darin bestand, dass ein menschlicher *Wizard* Hilfestellungen an die Studierenden senden konnte. Die ProbandInnen wussten in der Regel nicht, dass die Wizard-Nachrichten direkt von einem Menschen verschickt wurden. Durch eine quantitative und qualitative Analyse ermittelten die Autoren, dass das adaptive Feedback zu einem besseren konzeptionellen Verständnis, einer höheren Motivation und besseren Kollaboration führte. In der qualitativen Untersuchung zur Bestimmung der Effektivität des Wizards (Tsovaltzi et al., 2008) nutzen sie u. a. die Kategorien *gute Reaktion auf den Wizard* (z. B. Verbesserung ihrer Tätigkeit nach einer Nachricht) und *schlechte Reaktion auf den Wizard* (z. B. Nachricht zeigte keinen Effekt).

Der Forschungsstand von Tutorensystemen für Physical-Computing-Aktivitäten. In dem Bereich des Physical Computing wurden bislang wenige Untersuchungen vorgenommen, die zum Ziel hatten, Physical-Computing-Aktivitäten durch Technologien zu unterstützen. Spikol et al. (2016) skizzieren einen Ansatz und schlagen *Learning-Analytics*-Methoden vor, um den Problemlöseprozess des Physical Computing zu unterstützen. Davon sollen sowohl die SuS als auch die Lehrkräfte

profitieren. Des Weiteren werden verschiedene Hürden vorgestellt, die die Entwicklung von Tutorensystemen für Physical-Computing-Aktivitäten erschweren. Das können z. B. die Bildverarbeitung und die anschließende Interpretation sein, um auf das Verhalten der SuS schließen zu können. Ruiz et al. (2017) verwenden einen *Augmented-Reality*-Ansatz, in dem ProbandInnen bei der Konstruktion von Schaltkreisen Feedback erhalten. Mit Hilfe eines Tablets können sie die Korrektheit ihrer Schaltungen überprüfen und sich die korrekte Lösung ansehen. Erste Untersuchungen zeigten das Potential, dass weniger schwerwiegende Fehler bei den TeilnehmerInnen auftreten, wenn sie im Gegensatz zur Verwendung von analogen Schaltplänen die Augmented-Reality-Umgebung nutzten.
Kognitive Tutorensysteme werden insbesondere dadurch motiviert, dass bereits Studien zur Nutzung des Programms LOGO Abhängigkeiten (vgl. Kapitel 2) zwischen der Effektivität von Instruktionen für die Lernenden zu den gegebenen Hilfestellungen der Lehrkraft gezeigt wurden (vgl. Lehrer et al., 1989). Mit kognitiven Tutoren ist es hingegen möglich, die Hilfestellungen adaptiv und nachvollziehbar zu gestalten. Koedinger und Anderson (2013) entwickelten Guidelines für die Überführung von kognitiven Modellen in kognitive Tutorensysteme.
Aufgrund der aufgezeigten Forschungsergebnisse wird im Rahmen dieser Dissertation u. a. eine Hilfestellung entwickelt, die ein gestuftes Feedback erteilt und dabei zunehmend konkreter ein vorliegendes Problem beschreibt bzw. Lösungsansätze umfasst. Der Zeitpunkt des Feedbacks wird von betreuenden Personen festgelegt, die die SuS beim Lösen von Aufgaben beobachten. Sie schätzen den Zeitpunkt und die Intensität des Feedbacks ein und agieren regelgeleitet.

5.2 Methoden und Instrumente II

Dieser Abschnitt beschreibt Methoden und Instrumente, die der Untersuchung der 3. Forschungsfrage dienen (Phase Design & Construction III). Damit ergänzen sie die bereits in Kapitel 3.2 beschriebenen Werkzeuge. Darüber hinaus werden Test- und Hilfsinstrumente beschrieben, die zur quantitativen und qualitativen Beurteilung von Studien genutzt werden.

5.2.1 Methoden II

Ergänzend zu den aufgezeigten naturwissenschafts-didaktischen Methoden, wird in diesem Abschnitt die Wizard-of-Oz-Studie betrachtet.

Wizard-of-Oz-Studien

Eine sogenannte Wizard-of-Oz-Studie ist in der Forschung von Mensch-Maschine-Interaktion eine gängige Methode, um den Umgang von NutzerInnen mit Technologien zu untersuchen. Sie wird einerseits eingesetzt, um Prototypen vor der vollständigen Implementation zu testen, andererseits können qualitative und quantitative Daten zu Verhaltensmustern und Präferenzen von TeilnehmerInnen aufgenommen werden (Hoysniemi und Read, 2005).
Der Begriff geht historisch auf das Kinderbuch *The Wizard of Oz* zurück. Darin wurde davon ausgegangen, dass die Hauptfigur ein Zauberer ist, jedoch saß eine weitere Person hinter einem Vorhang, die Geschehnisse hervorrief. Dieses Prinzip wird in einer Wizard-of-Oz-Studie auf eine (vermeintlich) automatisierte Technologie übertragen, die jedoch von einem Menschen im Hintergrund gesteuert wird und nach außen ein intelligentes System zu sein scheint (Fraser und Gilbert, 1991).
Höysniemi und Read nehmen ein Literaturreview verschieden angelegter Wizard-of-Oz-Studien (WoZ-Studien) vor, die mit Kindern durchgeführt wurden. In ihrer Taxonomie stellen sie eine Vielzahl von Variationsmöglichkeiten innerhalb dieser Methode heraus. Für diese Arbeit relevante Möglichkeiten, werden in diesem Abschnitt vorgestellt. Zunächst kann der Umfang der eingesetzten Technologien durch einen *low-Tech* sowie einen *no-Tech* Prototypen realisiert werden (*Functionality-of-the-Technology*). Der Wizard agiert regelgeleitet (*Discretion-of-the-Wizard*), wobei er versteckt oder auch sichtbar agieren kann (*Visibility-of-the-Wizard*). Das heißt, wenn der Wizard nicht sichtbar ist, werden die ProbendInnen davon ausgehen, dass die Technologie ein voll funktionsfähiges System ist, dass ohne ein Eingreifen vom Menschen (*User Knowledge*) funktioniert. Ist der Wizard sichtbar, so wird dessen Rolle den ProbandInnen zuvor erläutert. Zwischen diesen beiden Extremen gibt es jedoch verschiedene Abstufungen und die Anzahl der Wizards kann ebenfalls variieren (*Number-of-Wizards*). Handelt es sich um mehrere Wizards, kann davon ausgegangen werden, dass das System möglichst unabhängig von den Wizards ist (Höysniemi et al.,

2004). Dafür ist es besonders wichtig, die an der Studie beteiligten Wizards zu schulen, damit eine bestimmte Aktion des Subjekts (hier die SuS) die gleichen Reaktionen bei allen Wizards hervorruft (Salber und Coutaz, 1993). Das Vorwissen bezüglich der getesteten Umgebung kann stark variieren (*Wizard Knowledge*), daher ist es legitim, WoZ-Studien für die Ermittlung des Umgangs mit dem System bei Novizen wie Experten zu analysieren. Eine Verknüpfung mit anderen Methoden, wie einem Interview, nach dem Umgang mit der Technologie ist ebenfalls eine bisher erprobte Möglichkeit. Diese sollte u. a. der Aufklärung der Studie dienen und klären, ob das Feedback tatsächlich voll automatisiert gegeben wurde (Dahlbäck et al., 1993). Neben der Erläuterung des Studienablaufs ist auch das Ziel der Studie zu erläutern. Im Allgemeinen gehen mit dieser Methode ebenfalls ethische Bedenken einher, insbesondere wenn mit Kindern gearbeitet wird (Hoysniemi und Read, 2005). Die Transparenz der Studie kann in diesem Fall im Vorfeld nicht gewährleistet werden, da es die Studienergebnisse beeinflussen würde. Es kann erst nach der Studie aufgeklärt werden, welche Personen zu welchem Zweck die ProbandInnen beobachteten. Deshalb sollten mögliche Folgen vor der Durchführung einer Wizard-of-Oz-Studie diskutiert und berücksichtigt werden, damit z. B. bei der Arbeit mit Kindern nicht falsche Vorstellungen über technische Geräte vermittelt werden.

5.2.2 Instrumente II

Mayer (2007) stellt im Rahmen von Methoden der Forschung zur wissenschaftlichen Erkenntnisgewinnung ebenfalls verwendete Testinstrumente dieses Bereichs vor. Etabliert sind schriftliche Tests, computerbasiertes Testen und das praktische Experimentieren. Er erläutert, dass oftmals praktische Experimente vorgezogen werden, da die erhobenen Leistungen mittels *Paper-and-Pencil*-Tests nicht mit denen des durchgeführten Experiments korrelieren (vgl. Roberts und Gott, 2004). Aufgrund dieser Ergebnisse wird für die folgende Untersuchung ein Zusammenspiel aus praktischen Aufgaben (Physical-Computing-Aufgaben), mit anschließender Videoanalyse (Codiermanuale) und *Paper-and-Pencil*-Tests (Test zu algorithmischen Strukturen und Physical-Computing-Test) verwendet. Ein Überblick darüber wird in diesem Kapitel gegeben.
Neben den Aufgabenstellungen wird ebenfalls durch diverse Hilfestellungen innerhalb der Studien interveniert. Sie werden kurz beschrieben und sind beispielhaft im Anhang enthalten.

Physical-Computing-Aufgaben II

Die Aufgaben der Studien 5 (siehe Tabelle 5.2) und 6 (siehe Tabelle 5.3) sind im Folgenden dargestellt. Es handelt sich bei den Aufgaben ausschließlich um den Teil der Studie, bei denen der/die Wizard/s Hilfestellungen gibt/geben und welche dokumentiert und anschließend evaluiert werden. Die Aufgaben wurden speziell für diese Untersuchung entwickelt. Dabei wurde darauf geachtet, dass sie die in Kapitel 4 beschriebenen Problemursachen

adressieren, da auch die folgenden Hilfestellungen von den Problemursachen abgeleitet werden. Die Aufgaben 1, 3 und 4 wurden bereits in den vorherigen Studien beschrieben.

Tabelle 5.2: Aufgaben bei der Arbeit mit LEGO Mindstorms-Robotern II

Aufgabenstellung	Verwendung in WoZ I	Verwendung in WoZ II
1. Erprobt, welche physikalischen Grenzen der Ultraschallsensor besitzt.	x	x
2. Der Ultraschallsensor wird nun seitlich (90 Grad) zur Fahrtrichtung ausgerichtet. Der Roboter soll so programmiert werden, dass er den Eingang einer Wand findet, an der er vorbeifährt. Auf Höhe des Eingangs soll der Roboter stehen bleiben.	x	x
2.1 Programmiert den Roboter so, dass er erneut an der Wand entlangfährt und in den Eingang hineinfährt.	x	x
2.2 Der Eingang wird nun verkleinert. Programmiert den Roboter, damit er die Aufgabe 2.1 weiterhin erfüllt.	x	x
3. Der Ultraschallsensor wird nun nach vorne in Fahrtrichtung ausgerichtet. Programmiert den Roboter so, dass er um eine Kiste fahren kann, alleinig unter der Nutzung des Ultraschallsensors.	x	x
4. Programmiert den Roboter so, dass er in der Lage ist ein vorgegebenes Labyrinth zu durchfahren.	x	
4.1 Verändert die Startposition des Roboters und testet euer Programm erneut. Nehmt gegebenenfalls Änderungen an eurem Programm vor, damit der Roboter das Labyrinth durchfahren kann.	(x)	
4.2 Programmiert den Roboter so, dass er anhält, sobald er das Labyrinth verlassen hat.	(x)	

Die Aufgaben, die mit einem (x) gekennzeichnet sind, wurden aus zeitlichen Gründen nicht von allen Gruppen bearbeitet.

Neu in diesen Studien sind die Aufgaben 2, 2.1 und 2.2, die die Abtastrate von Sensoren adressieren. Die Herausforderung dabei ist, dass die Software genau an die Konstruktion der Hardware angepasst werden muss und die SuS die Abfrage des Sensors geschickt in das Programm implementieren müssen. Es muss beachtet werden, welche Zweige im Programm abgearbeitet werden, bis der Sensorwert erneut ermittelt wird und Aktionen auslösen kann.

Aufgabe 4.1 und 4.2 dienen der Differenzierung für besonders zügige SuS. In der Aufgabe 4.1 soll das Programm erneut getestet werden, damit die Einsetzbarkeit des grundlegenden Algorithmus für weitere Anwendungsfälle gezeigt werden kann. Kleine Veränderungen der Umgebung können dazu führen, dass das ursprüngliche Programm nicht mehr zu dem gleichen Ergebnis führt. Mit einigen Adaptionen im Drehwinkel und der Distanz des Fahrens ist es möglich, einen Algorithmus zu schreiben, der für viele Labyrinthe und Ausgangslagen funktioniert. Haben die SuS bereits für Aufgabe 4 häufige Sensorabfragen und kleine Bewegungsveränderungen des Roboters implementiert, so kann möglicherweise das identische Programm für Aufgabe 4.1 zur Lösung führen. Zum Lösen der Aufgabe 4.2 ist ein

wichtiger Aspekt, dass das Programm auf dem Roboter beendet wird, sobald der Ausgang des Labyrinths gefunden wurde. Das kann durch die Implementation eines weiteren Schalters gelöst werden, der das Programm beendet, sobald der Ultraschallsensor einen Wert $> 2,00\,m$ misst. Dieser Wert kann ausschließlich außerhalb des Labyrinths gemessen werden. Die SuS müssen sich dabei überlegen, wie sie Endlosschleifen unterbrechen können.

Tabelle 5.3: Aufgaben bei der Arbeit mit Nao-Robotern

Aufgabenstellung	Verwendung in	
	WoZ I	WoZ II
1. Programmiert den Roboter so, dass er in der Lage ist einen Ball zu greifen. Arbeitet dabei nur mit dem oberen Teil des Körpers.	x	x
1.1 Wählt nun einen kleineren Ball und testet das Programm erneut. Nehmt gegebenenfalls Änderungen am Programm vor.	x	x
2. Der Roboter soll nun einen Stift festhalten. Schreib ein Programm, so dass der Roboter in der Lage ist ein Kreuz auf einem Blatt Papier zu zeichnen. Der Stift darf dem Roboter gereicht werden und es steht eine Kiste als Unterlage zur Verfügung.	x	x
2.1 Programmiert nun noch weitere Symbole oder Buchstaben, die der Roboter zeichnen soll.	x	
2.2 Schreibt ein Programm, das dem Roboter ermöglicht auf verschiedene Sprachbefehle unterschiedliche Symbole zu zeichnen.	x	
3. Programmiert die Kopftasten am Roboter so, dass er beim Drücken der vorderen Taste aufsteht und sich beim Drücken der mittleren Taste in die Rest-Pose begibt.	x	x
3.1 Benutzt das Programm von 3. und erprobt das vorgefertigte Programm zur Ballsuche. Der Roboter soll auf den Ball zeigen, wenn er ihn gefunden hat. Wann erkennt der Roboter den roten Ball?	x	x
3.2 Aufbauend auf 3.1 soll nun ein Programm geschrieben werden, mit dem der Roboter einen Ball im Raum suchen kann und zu diesem hinläuft.	x	(x)

Die Aufgaben, die mit einem (x) gekennzeichnet sind, wurden aus zeitlichen Gründen nicht von allen Gruppen bearbeitet.

Im Umgang mit dem Nao-Roboter setzen sich die SuS ebenfalls mit Kontrollstrukturen, Input/Output und seriellen/parallelen Prozessen auseinander. Die SuS wissen bereits, wie sie mit der Software Choregraphe (vgl. Abschnitt 3.2.2) *Keyframes* aufnehmen. Das heißt, sie können den Roboter durch die äußere Veränderung der Motoren in eine Position bringen und diese speichern. Dabei wird die Stellung aller Motoren aufgenommen und die Position kann als Box in dem Programm gespeichert werden. Beim Aufruf der Box nimmt der Roboter die aufgenommene Position ein.
In den Aufgaben 1, 1.1 und 2 handelt es sich zunächst um einen reinen Output, bei dem die SuS Keyframes sorgfältig aufnehmen und diese anschließend miteinander zu einem Programm verbinden. Eine besondere Schwierigkeit steckt darin, dass die Motoren, die an dem Roboter die Schultergelenke bilden, ungenau sind und dadurch die Arme

schwer zusammenzuführen sind. Bei der 2. Aufgabe muss hingegen die Umgebung stärker berücksichtigt werden, das heißt die Lage und Stabilität des Zeichenuntergrunds (Papier), damit das Zeichnen mit einem Stift möglich ist. In der Aufgabe 2.2 werden Boxen genutzt, die die Eingabe von Sprachbefehlen ermöglichen. Dafür wird von den SuS ein Input generiert und sie müssen herausfinden, wie sie sprechen sollten (Stimmlage und Position zum Sensor), damit eine Eingabe erfolgreich ist. In Aufgabe 3 wird ein Input über die Kopftaste generiert, was zunächst weniger fehleranfällig ist als eine Spracherkennung. Nun wird der untere Teil des (Roboter-)Körpers einbezogen, was insbesondere bei der Aufnahme von Keyframes bedeutet, dass auf die Position von vielen Motoren geachtet werden muss. Steht der Roboter leicht zu einer Seite geneigt, so kann er bei Folgebewegungen fallen. Bei der Anwendung der Ballsuche in Aufgabe 3.1 sind insbesondere Umweltfaktoren zu berücksichtigten. Die Box wird vorprogrammiert, einen roten Ball zu erkennen. Einerseits muss ermittelt werden, welche Farbe als Rot erkannt wird, andererseits sollten Umweltfaktoren durch rote Kleidung o. Ä. berücksichtigt werden. Aufgabe 3.2 adressiert die Entwicklung eines geeigneten Suchalgorithmus, der einen Ball im Raum finden kann. Dafür sind gezielte Kopfbewegungen des Nao-Roboters notwendig.

Unterstützungen für die SuS in den Studien

Basierend auf den theoretischen Grundlagen wurde bereits gezeigt, dass die Forschung zur wissenschaftlichen Erkenntnisgewinnung die Nutzung diverser Hilfestellungen empfiehlt, um den Problemlöseprozess zu unterstützen. Im Rahmen dieser Arbeit werden drei verschiedene Hilfestellungen entwickelt, die unterschiedliche Aufgaben erfüllen. Hilfestellung *4-Phasen analog* dient lediglich als Leitfaden, wie ein Problemlöseprozess im Physical Computing strukturiert werden kann. Die Hilfestellungen *Evaluationsphase digital* und *Problemtaxonomie gestuft* hingegen intervenieren konkret beim Problemlösen, um den Prozess zu unterstützen und die Auswirkungen des Instruments zu evaluieren. Die Unterstützungen werden im Folgenden vorgestellt.

Beschreibung der Hilfestellung *Evaluationsphase digital* Die 3. Studie verfolgt das Ziel den Physical-Computing-Prozess verstärkt zu strukturieren, indem besonders auf die Evaluationsphase eingegangen wird. In einem einführenden Text ist beschrieben, welche Phasen bei der Erkenntnisgewinnung üblicherweise durchlaufen werden. Anschließend werden die SuS dazu aufgefordert, bei jedem Durchlauf des Prozesses Notizen anzufertigen. Dieses Arbeitsblatt wird digital zur Verfügung gestellt und fordert dazu auf, die Evaluation detaillierter vorzunehmen. Dazu wird auf dem Arbeitsblatt (siehe Anhang B.2) zwischen verschiedenen Problemursachen unterschieden, die in der Evaluation berücksichtigt werden sollen. Durch die Vorstrukturierung werden die SuS dazu aufgefordert, in jeder Evaluationsphase mögliche Fehler innerhalb der Teilgebiete Software, Hardware und Umgebungseinflüsse einzuordnen. Sie sollen versuchen zu jedem Gebiet einen möglichen Fehler für

jede Evaluationsphase zu ermitteln. Anschließend sollen sie entscheiden, welche Problemursache in diesem Fall für wahrscheinlich erachtet wird und anschließend weiterverfolgt werden sollte.

Alle Gruppen der 3. Studie füllten diese Hilfestellung aus. In Anhang C.4 ist das Arbeitsblatt einer Gruppe exemplarisch dargestellt.

Beschreibung der Hilfestellung *Problemtaxonomie gestuft* Die Hilfestellung *Problemtaxonomie gestuft* wird basierend auf den Ergebnissen der 2. Forschungsfrage (Kapitel 4) entwickelt. Diese beschäftigt sich mit konkreten Problemen und den zugehörigen Problemursachen, die während der Arbeit mit Physical-Computing-Geräten auftreten. Abgeleitet von den Videodaten wurden verschiedene Ebenen des Feedbacks ermittelt, die den SuS in diesem Fall gegebenenfalls weiter geholfen hätten. Das mehrstufige Feedback, welches immer genauer wird, hat sich im Bereich von intelligenten Tutorensystemen etabliert und bewährt (vgl. Kapitel 5.1). Die ProbandInnen bekommen zum Beginn der Studien die Problemtaxonomie erklärt, damit sie anschließend durch ein Zeigen auf einen Bereich wissen, wie dieser definiert ist.
In der *1. Ebene* der entwickelten Hilfestellung erhalten die SuS ein Feedback darüber, um welche Problemursache es sich handelt. Dazu bekommen sie lediglich den Namen der Problemursache genannt, was Hardware, Software, Umgebung oder mathematische/physikalische Grundlagen bezeichnen kann. In dieser Ebene können ebenfalls die Überschneidungen von zwei oder drei Bereichen adressiert werden (Hardware/Software, Hardware/Umgebung, Umgebung/Software, System).
Die *2. Ebene* umfasst eine weitere Einschränkung des Problembereichs, welche wie folgt in Unterkategorien unterteilt werden kann (siehe Tabelle 5.4; vgl. Erläuterungen in Kapitel 4): Sofern in der 1. Ebene eine Schnittstelle von mehreren Problemursachen genannt wurde, sollen in der 2. Ebene die Unterkategorien aus allen aufgezeigten Fehlerquellen genannt werden.
In der *3. Ebene* sollen die vorgegebenen Kategorien detaillierter und bezogen auf das vorliegende Problem beschrieben werden, ohne die Lösung zu nennen. Beispielsweise haben die SuS einer Variable einen falschen Wert zugewiesen und müssen diesen ändern. Innerhalb dieser Ebene ist z. B. einzuordnen, wenn die SuS auf das Verändern einer Variable hingewiesen werden, ohne einen geeigneten Wert für die Variable zu nennen. Die 3. Ebene kann gegebenenfalls übersprungen werden, abhängig von dem konkreten Problem. In einigen Bereichen der Taxonomie bietet sich keine weitere Unterteilung an, z. B. bei dem Problembereich Umgebung. An dieser Stelle ist der Hinweis, dass die Lichtverhältnisse ein Problem darstellen, ausreichend und kann möglicherweise nicht genauer eingegrenzt werden, u. a. da konkrete Werte immer situationsspezifisch sind.
Erst in der *4. Ebene* soll das Problem explizit benannt und eine Musterlösung erläutert werden. Es sollte ein konkreter Wert, der zur Lösung führt, behandelt werden, z. B. „der

Tabelle 5.4: Identifizierte Haupt- und Unterkategorien von Problemursachen während des Physical-Computing-Prozesses

Hauptkategorien	Unterkategorien
Hardware	Konstruktion (1a) Fehlfunktion von Sensoren (1b) Defekter Sensor (1c)
Software	Programmcode (2a) Programmierumgebung (2b) Firmware des Roboters (2c)
Umgebung	Natürliche Umgebungseinflüsse (3a) Menschlicher Eingriff in die Umgebung (3b)
Mathematische/Physikalische Grundlagen	Verwendung von Operatoren (4a) Physikalische Funktion von Sensoren (4b) Konstruktion eines physikalischen Experiments (4c) Bestimmung von Grenzwerten (4d)

linke Motor sollte auf eine 30-Grad-Drehung eingestellt werden“.

Nach der 5. Studie wurden kleine Veränderungen an dieser Hilfestellung vorgenommen. Einerseits sollte der Wizard die 1. und 2. Phase zunehmend überspringen, wenn es als Feedback ungeeignet erschien, da die SuS bereits diese Einschränkung vornahmen oder es offensichtlich war. Andererseits wurde innerhalb des Bereichs Software konkreter bezeichnet, welches Problem im Programmcode vorliegt, z. B. ob Boxen fehlten oder falsche Kontrollstrukturen verwendet wurden.

Test zu algorithmischen Strukturen

Das *Messinstrument zur Anwendung von imperativen Kontrollstrukturen* (vgl. Anhang A) wurde kürzlich entwickelt und erste Ergebnisse wurden publiziert (Mühling et al., 2015). Abgeleitet von der englischen Übersetzung wird der Test in dieser Arbeit auch kurz als *Basis-Programmierkompetenz-Test* bezeichnet. Es handelt sich dabei um ein validiertes Testinstrument, welches die Verwendung von Kontrollstrukturen bei Schülerinnen und Schülern von der 7. - 10. Klasse misst.
Der Test ist in sechs Aufgaben mit ansteigender Schwierigkeit gegliedert. In jeder Aufgabe ist ein Spielbrett mit Koordinaten und einem Kompass angegeben, auf dem ein Roboter steht. Dazu ist eine Erläuterung der Aufgabe gefolgt von einem Pseudocode dargelegt. Letzterer enthält Angaben zur Bewegung, die der Roboter auf dem Spielbrett ausführen soll. Die Schülerinnen und Schüler sollen als Lösung der Aufgaben angeben, auf welchem Feld der Roboter nach Ablauf des Programms steht und zu welcher Himmelsrichtung er

ausgerichtet ist. Werden das Feld und die Himmelrichtung richtig angegeben, so erhalten die SuS einen Punkt für die Aufgabe. Aufgabe 5 ist in zwei Teilaufgaben untergliedert. Daher ist die höchste mögliche Punktzahl für den Test mit sieben Punkten festgelegt.
Der Test wurde durch die *Item-Response-Theory* validiert und ist eindimensional gestaltet, so dass ein latentes Konstrukt durch mehrere Items ermittelt wird. Die inhaltlichen Bereiche innerhalb des Tests umfassen: Sequenzen von Operationen, Bedingungen, Schleifen mit fester Anzahl von Wiederholungen, Schleifen mit Abbruchbedingungen, und die Verschachtelung dieser Konstrukte. Das gemessene latente Konstrukt ist die *Fähigkeit zur Verwendung von Kontrollstrukturen.*
Bislang ist das Testinstrument nur bei den Autoren erhältlich, da noch verschiedene Adaptionen vorgesehen sind. Es unterliegt jedoch der *Creative Commons*-Lizenz (CC BY 4.0). In dieser Arbeit und den damit verbundenen Studien wird dieses Testinstrument verwendet, um den Leistungsstand der Teilnehmenden zu erheben und gegebenenfalls zu vergleichen. In der 4. - 6. Studie diente das Testinstrument als Pre- und Posttest, um den Zuwachs der Fähigkeit der Verwendung von Kontrollstrukturen zu ermitteln.

Physical-Computing-Test

Dieses Testinstrument (siehe Anhang A) wurde im Rahmen der vorliegenden Dissertation konzipiert, um die *Fähigkeit der Evaluation von auftretenden Problemen in Physical-Computing-Aktivitäten* zu erheben. Dabei richtet er sich an die Zielgruppe der 7. - 10. Klasse. Der Test ist in zwei Teile aufgeteilt: einer Selbsteinschätzung und den Aufgaben zur Messung von zwei latenten Konstrukten. Bei der Selbsteinschätzung im 1. Teil werden die Vorkenntnisse zur Informatik und zu verschiedenen Physical-Computing-Geräten auf einer Likert-Skala von 1 - 6 („sehr wenig“ bis „sehr viel“) erfragt.
Der 2. Teil des Tests enthält drei Aufgaben, wobei Aufgabe 2 und 3 jeweils ein latentes Konstrukt eindimensional erheben. In der 1. Aufgabe sollen die Schülerinnen und Schüler eine Einschätzung angeben, mit welcher Häufigkeit die drei Hauptproblemursachen (Software, Hardware und Umgebung) beim Lösen von Aufgaben mit Robotern auftreten. Hier wird eine Likert-Skala von 1 - 6 („stimmt gar nicht“ bis „stimmt sehr“) angewandt.
Die 2. Aufgabe misst das latente Konstrukt *EVA-Prinzip* in Form einer Multiple-Choice-Aufgabe. Es handelt sich um eine Zuordnung von Aktionen zu den Bereichen *Input, Processing* und *Output.* Der Bereich *Input* wird weiterhin in *analog* und *digital* unterschieden, wobei die Bewertung dieser Spezialisierung innerhalb des Inputs erst ab der 10. Klasse berücksichtigt werden sollte. Das Konstrukt wird anhand von 12 Items gemessen, die auf die drei Bereiche gleich verteilt sind. Bei dieser Aufgabe ist die Zuordnung für jede Aktion zu genau einem Bereich (Input, Processing oder Output) möglich. Es ist offensichtlich, dass eine Eingabe und Ausgabe nicht ohne eine Form der Datenverarbeitung vonstatten gehen kann, jedoch soll notiert werden, um welchen Teil des EVA-Prinzips es sich bei den Aktionen vorrangig handelt. Die SuS sollen den zutreffenden Bereich ankreuzen und wer-

den im Anschluss dazu aufgefordert, auf drei vorgedruckten Zeilen ihre Entscheidung zu begründen.
In der 3. Aufgabe (Freitextaufgabe) werden ein Zielzustand und das abweichende Verhalten eines Roboters beschrieben. Die erste Version des Tests enthielt zehn Items mit möglichen Erklärungen für das Verhalten des Roboters, die zu den Hauptproblemursachen (Software, Hardware und Umgebung) zugeordnet werden sollten. In der zweiten Version des Tests, wird ausschließlich vorgegeben, dass für jede der drei Hauptproblemursachen drei mögliche Probleme gefunden und notiert werden sollen, die das Verhalten erklären. Anschließend soll die Entscheidung erneut im Freitext begründet werden. Das gemessene latente Konstrukt dieser Aufgabe ist *die Unterscheidung der Komponenten eines Physical-Computing-Systems.*
Vor den Einsatz in der Studie, wurde dieser Test mit $N = 67$ SuS von verschiedenen Berliner Schulen pilotiert. Diese Stichprobe umfasst SuS der 6. - 12. Klasse, die an Physical-Computing-Arbeitsgemeinschaften oder einem Physical-Computing-Workshop teilnahmen, sowie ITG-Kurse (Informationstechnische Grundbildung), die zuvor mit Robotern gearbeitet hatten. Der Test wurde in die Testhefte A und B unterteilt, die sich lediglich in der Reihenfolge der Items unterschieden. Es stellte sich heraus, dass es kaum Unterschiede zwischen den Testheften A und B gab. Die Itemschwierigkeit der Aufgabe 2 lag zwischen Mittelwerten von 0.55 - 0.91, weshalb anschließend einige einfache Items entfernt und schwerere hinzugefügt wurden. Bei der Ermittlung des Mittelwerts wurden alle Items innerhalb einer Kategorie (Input, Output oder Processing) betrachtet. Die 3. Aufgabe wies Itemschwierigkeiten von 0.76 - 0.95 auf, wobei 70% der Items über einer Schwierigkeit von 0.90 lagen und damit zu einfach waren. Diese Mittelwerte wurden innerhalb der Kategorien Software, Hardware und Umgebung ermittelt. Nach der Eliminierung der extremen Altersunterschiede und der ausschließlichen Betrachtung von Ergebnissen der 9. Klassenstufe ($N = 24$), wurden Itemschwierigkeiten MW von 0.70 - 1.00 erreicht. Da bei diesen Testergebnissen bei einer Verwendung des Testinstruments zur Erhebung des Ist-Zustandes bzw. zum Messen einer Verbesserung als Pre- und Posttest keine großen Effekte sichtbar werden können, wurde dieses Item im Freitextformat designt. Aufgrund der Beobachtung von Problemen während des Physical-Computing-Prozesses innerhalb von Studien kann davon ausgegangen werden, dass die SuS weniger Probleme mit der Zuordnung der Problembeschreibungen zu den Kategorien haben. Schwieriger erscheint der Schritt, die Probleme zu beschreiben und mögliche Ursachen dafür zu finden. Deshalb werden nun die Kategorien der Problemursachen Software, Hardware und Umgebung vorgegeben, und die SuS geben für jede Kategorie jeweils drei Beispiele an.

Geräte und Software II

Nao-Roboter werden von der Firma Aldebaran hergestellt und haben sich weltweit als humanoide Roboter etabliert (siehe Abbildung 5.1). Aus diesem Grund werden einige

Ligen des RoboCup[1] ebenfalls mit Nao-Robotern durchgeführt. Die Nao H25-Roboter[2] sind mit zwei Kameras ausgestattet (eine nach vorne oben und die andere nach vorne unten ausgerichtet), um die Umgebung durch Bildverarbeitungsmethoden wahrzunehmen. Darüber hinaus verfügt der Roboter über eine Vielzahl von Drucksensoren, die z. B. in den Handflächen und unter den Füßen angebracht sind, um es dem Roboter zu ermöglichen, die Balance zu halten und zu erkennen, ob die Füße flach auf dem Boden stehen. Es sind Drucksensoren auf dem Kopf des Roboters angebracht, um dadurch z. B. Eingaben zu tätigen. Entsprechend der menschlichen Gelenke, werden die Extremitäten des Roboters durch Motoren bewegt. Von ihnen kann ebenfalls die Position abgerufen und damit die Gelenkstellung errechnet werden. Durch vier gerichtete Mikrophone und Lautsprecher verfügt der Roboter über eine Ton- und Sprachausgabe und kann Töne sowie Sprache als Eingabe verarbeiten. Über zusätzliche Module kann der Roboter mit Wi-Fi und Ethernet verbunden werden und kann somit zu anderen Robotern eine Verbindung aufbauen. Über diverse LED-Lampen kann der Roboter ein Feedback, z. B. über Prozesse, an die Umgebung zurückgeben. Diese sieben Komponenten (Bewegung, Fühlen, Hören und Sprechen, Sehen, Verbindung, Denken) sollen dazu beitragen, ähnliche Daten, wie von menschlichen Sinnen durch den Roboter aufnehmen zu können und ihn natürlich interagieren zu lassen (SoftBank Robotics, 2017). Die Nao-Roboter werden seit 2006 produziert und weiterentwickelt, so dass sie seit einigen Jahren sogar frei zu erwerben sind.

Abbildung 5.1: Nao-Roboter

In den Studien wird eine visuelle Programmierumgebung namens *Choregraphe*[3] (Version 2.1.3) von Aldebaran genutzt, wie in Abbildung 5.2 verdeutlicht. Diese Sprache ist blockbasiert aufgebaut, wobei die Blöcke frei auf einem Feld angeordnet und anschließend

[1] http://www.robocup.org
[2] http://doc.aldebaran.com/2-1/family/nao_h25/index_h25.html
[3] https://community.ald.softbankrobotics.com

verbunden werden. Außerdem umfasst das Programm einen Simulator (unten rechts), bei dem Aktionen und Interaktionen mit dem Roboter zuerst simuliert werden können und anschließend das gleiche Programm auf dem Roboter ausgeführt werden kann. Die ausschließliche Verwendung des Simulators ist ebenfalls möglich. Auf der linken unteren Seite sind verschiedene Bibliotheken mit Aktionen enthalten. Die vorgefertigten Blöcke in dem Programm können durch eine Adaption in den Python-Scripten verändert werden. So können ebenfalls neue Blöcke erzeugt werden. Somit ist es möglich, eigene Bibliotheken zusammenzustellen und für die SuS einen besseren Überblick über zu benutzende Blöcke zu erstellen.

Durch Choregraphe ist es möglich, an dem physischen Roboter die Extremitäten zu bewegen und diese Positionen aufzuzeichnen und in Blöcken zu speichern. Die Geschwindigkeit, in der der Roboter von einer zur nächsten Pose geht, kann ebenfalls adaptiert werden. Diese Funktion der Aufnahme von Bewegungen wird als Erstellung von *Keyframes* bezeichnet. Enthaltene Standardblöcke sind z. B. Bewegungen, wie Aufstehen und Hinsetzen des Roboters. Aktionen zum Greifen eines Balls können mit Keyframes realisiert werden, indem die Arme des Roboters parallel nach vorne gehalten und als eine Bewegung aufgenommen werden. Als zweite Bewegung werden die Arme so nach vorne gehalten, dass sich die Hände des Roboters berühren. In Choregraphe können diese Blöcke anschließend miteinander zu einer Bewegung verbunden werden, wobei zunächst die Bewegung des ersten Blocks und dann die des zweiten Blocks ausgeführt werden.

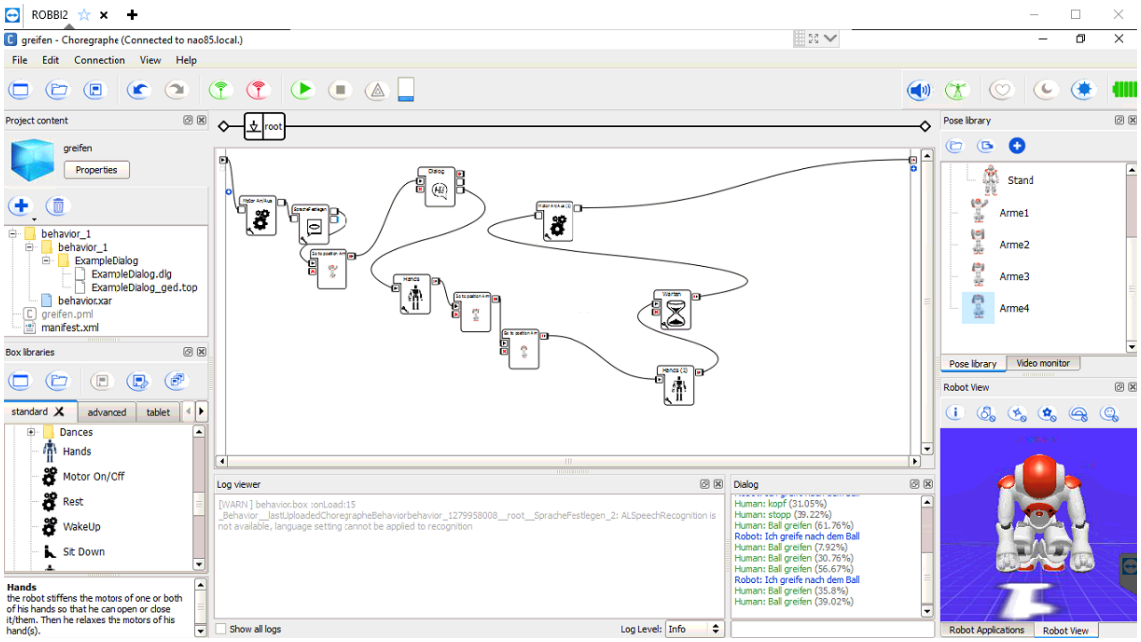


Abbildung 5.2: Programmierumgebung Choregraphe

Slack ist ein Messenger-Dienst[4], der genutzt werden kann, um Nachrichten an verschiedene Gruppen zu versenden. Dieser Dienst soll insbesondere kollaborative Arbeitsprozesse unterstützen. Da er webbasiert funktioniert, ist lediglich eine Anmeldung über E-Mail-Adressen (im Gegensatz zur Anmeldung mit Telefonnummern) erforderlich. Das eignet sich besonders für die temporäre Nutzung zur Durchführung von Studien. Es wurde die Version 3.23.1 verwendet, die zu diesem Zeitpunkt die Aktuellste war. In Abbildung 5.3 ist die beispielhafte Nutzung des Dienstes in der Wizard-of-Oz-Studie II illustriert. Auf der linken Seite sind einzelne Gruppen abgebildet, in denen Nachrichten versandt werden können. Auf der rechten Seite sind alle Nachrichten abgebildet, die eine Gruppe erhalten hat. Nach Abschluss der Intervention einer Gruppe werden die versandten Nachrichten gelöscht, damit die anderen Gruppen nicht vorherige Hilfestellungen nutzen können.

[4] https://slack.com/intl/de-de

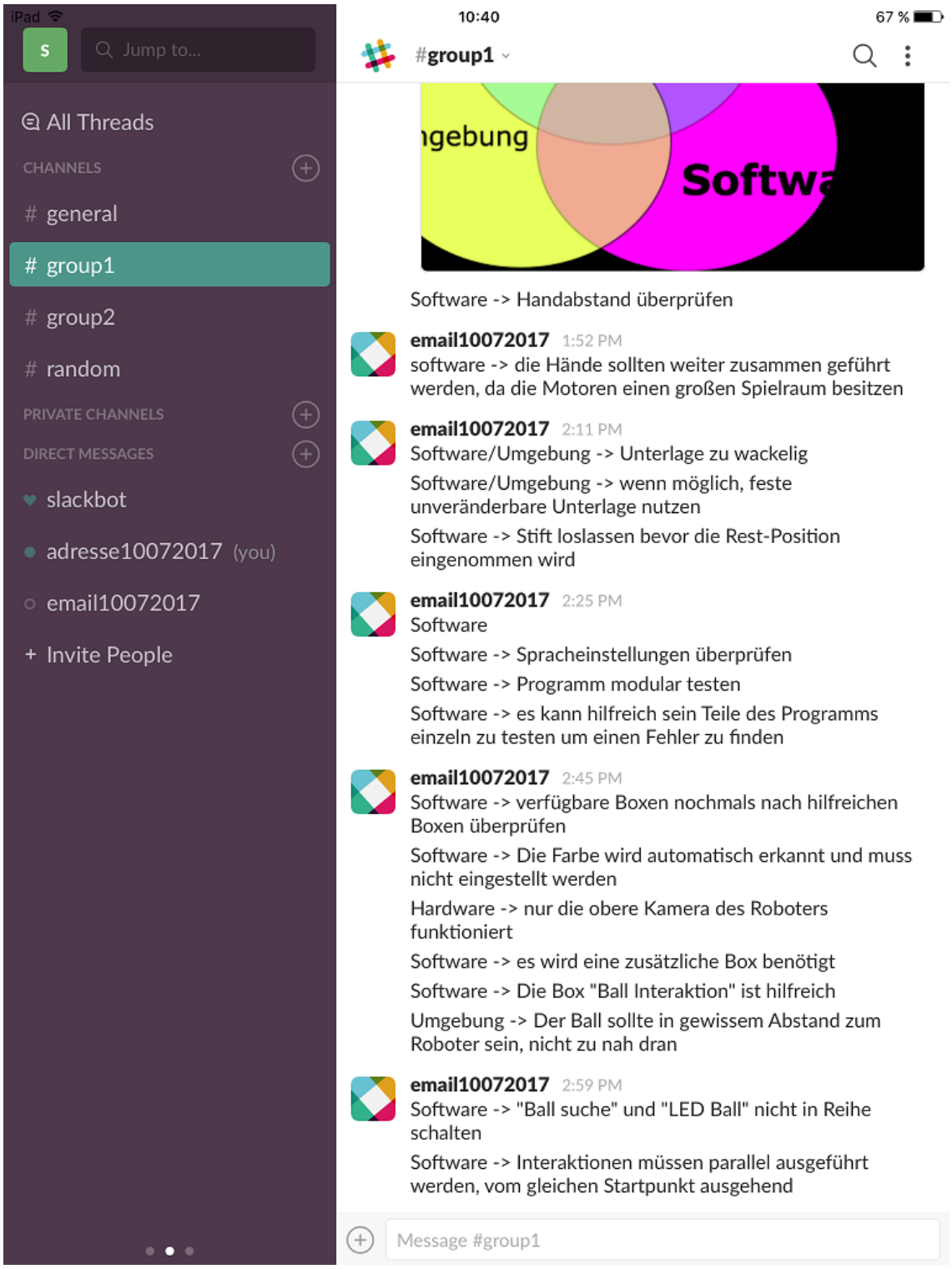


Abbildung 5.3: Messenger-Dienst Slack

5.2.3 Ablauf der Datenerhebung II

In den Studien 5 und 6 sollten die SuS keine expliziten Fragen stellen und die betreuenden Personen nur eingreifen, wenn sie beobachten, dass die SuS Probleme haben oder Fehler

machen, die sie nicht selbstständig lösen oder lösen können. Bei der 5. Studie wurden die Gruppen aufgeteilt, so dass maximal drei Gruppen gleichzeitig in einem Raum die Aufgaben bearbeiteten. Die anschließenden Interviews wurden mit allen Gruppen einzeln in einem gesonderten Raum durchgeführt. In der 6. Studie waren zwei betreuende Personen anwesend, darunter die Versuchsleiterin und ein Experte im Umgang mit Nao-Robotern, für den Fall, dass diese neu gestartet werden müssen oder andere Probleme mit den Robotern auftreten.
Die 6. Studie war technisch am aufwendigsten. Die ProbandInnen wurden durch eine Kamera und ein gerichtetes Mikrofon in Raum aufgezeichnet, wobei das Bild in Echtzeit zu dem Wizard übertragen wurde. In Abbildung 5.4 ist der technische Aufbau und die Ansicht für den Wizard dargestellt. Auf den (im Bild oberen) Bildschirm wurde der Arbeitsplatz der SuS übertragen, so dass deren Interaktion mit den Robotern sichtbar war. Die Unterhaltung der SuS war zum Teil für den Wizard zu hören, da er im Nebenraum saß und die Verbindungstür angelehnt wurde. Der Bildschirm der SuS wurde durch Team Viewer auf den (linken) Laptop des Wizards übertragen. Dadurch konnte der Wizard konkrete Veränderungen sehen, die die SuS am Programm vornahmen und ihre Körpersprache mittels der Kamera beobachten. Auf einem weiteren Laptop (rechts im Bild) hatte der Wizard den verwendeten Messenger-Dienst Slack geöffnet, mit dem er Nachrichten an die SuS sandte und ihre Reaktionen mitverfolgen konnte.

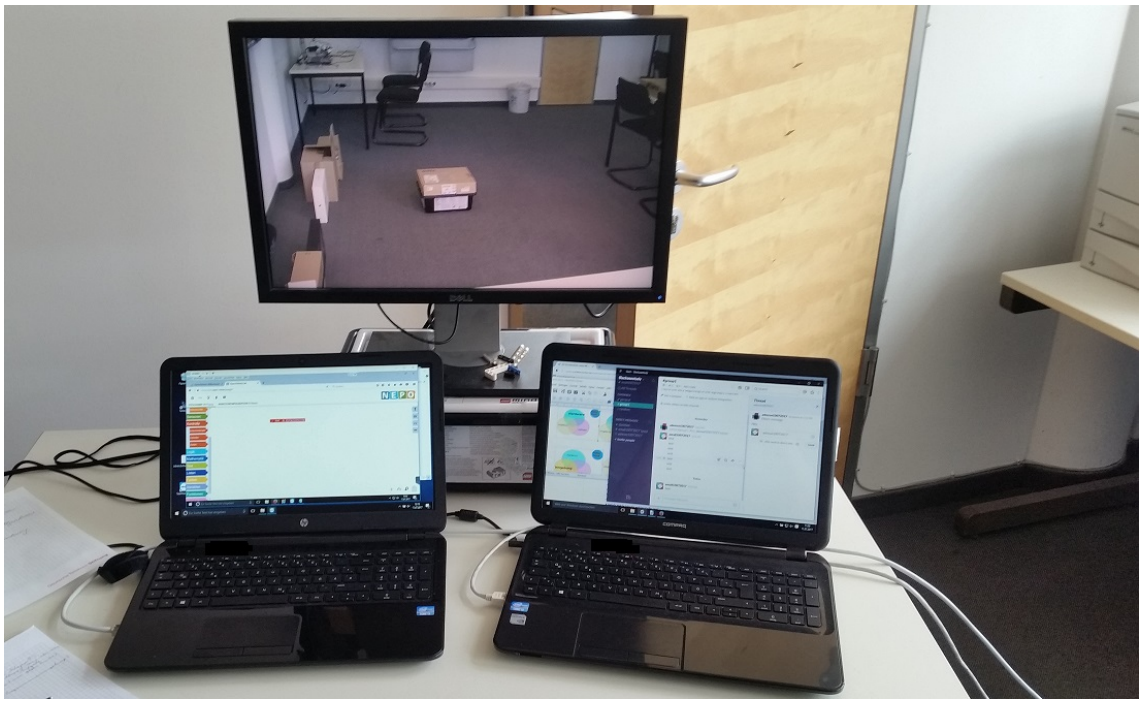


Abbildung 5.4: Aufbau der Wizard-of-Oz-Studie II, Sicht des Wizards

5.3 Teilstudie zur Intensivierung der Evaluationsphase

Wie bereits in den methodischen Grundlagen (vgl. Kapitel 5.2.2) beschrieben, wird die Evaluationsphase teilweise übersprungen (Patzwaldt et al., 2014). Ein Überspringen dieser Phase wurde ebenfalls bei der Untersuchung des Physical-Computing-Prozesses (vgl. Kapitel 3) festgestellt. Aus diesem Grund wird ein Ansatz verwendet, um die Evaluationsphase zu intensivieren. Das Auslassen von Prozessphasen wird betrachtet, da es im Zusammenhang mit Problemen beim Experimentieren stehen kann (Walpuski und Sumfleth, 2007).
Patzwaldt et al. (2014) beobachteten, dass die kognitiven Prozesse der Studierenden parallel zu den erteilten Teilaufgaben abliefen, die den Inquiry-Prozess strukturierten. Deshalb wurde den SuS in dieser Studie ebenfalls ein Teil der Evaluationsphase explizit vorstrukturiert. Sie hatten die Aufgabe, die verschiedenen Problemursachen des Physical-Computing-Prozesses in jeder Evaluationsphase zu betrachten. Dabei sollten sie für jede Kategorie Problemursachen (Hardware, Software und Umgebung) analysieren, die in ihrer konkreten Situation mögliche Probleme innerhalb der Kategorien sein können. Anschließend war zu entscheiden, in welcher Kategorie sie das Problem beheben wollen. Ähnlich wie bei dem Programm Scratchpad zur Auswahl von zu testenden Hypothesen von van Joolingen und de Jong (1993), wurden in dieser Studie die möglichen Problemursachen und Lösungen von den SuS in einem digitalen Arbeitsblatt angeordnet, auf dessen Basis sie ihr weiteres Vorgehen planten. Das digitale Arbeitsblatt (Hilfestellung *Evaluationsphase digital*) wurde den SuS als Word-Dokument zur Verfügung gestellt und sollte auf dem gleichen Computer ausgefüllt werden, an dem die Programmierung des Roboters vorgenommen wurde. Die SuS bekamen zum Beginn der Intervention die Anweisung das digitale Arbeitsblatt nach jeder Ausführung ihres geschriebenen Programms (Durchführungsphase) auszufüllen.
In diesem Abschnitt werden lediglich die Studien miteinander verglichen, die mit LEGO Mindstorms-Robotern durchgeführt wurden. Da bereits in den Kapiteln 3 und 4 deutlich wurde, dass der Physical-Computing-Prozess und auftretende Probleme bei der Arbeit mit unterschiedlichen Geräten verschiedene Schwerpunkte haben können, wird nun die Form der Unterstützung verändert, nicht aber das Gerät, mit dem gearbeitet wurde. Außerdem erhöht dies die Vergleichbarkeit der Studien untereinander.
Die genutzte Vorlage ist in Anhang B.2 zu finden, während eine beispielhafte Nutzung der Hilfestellung von Gruppe AB der 3. Studie in Anhang C.4 abgebildet ist. Weitere Studienbedingungen sind dem Kapitel 1.3 zu entnehmen. Das Ziel dieser Untersuchung ist es, zu ermitteln, ob durch die Hilfestellung *Evaluationsphase digital* der Verlauf des Physical-Computing-Prozesses verändert wird. Es gilt zu überprüfen, ob die SuS weniger Evaluationsphasen im Vergleich zur 1. und 2. Studie überspringen. Anhand der Vielfältigkeit an betrachteten Problemen und der Lösungsansätze in den Problemursachen, soll darüber hinaus die Qualität der Evaluationsphase untersucht werden.

5.3.1 Analyse quantitativer Daten

Die quantitative Analyse der Hilfestellung *Evaluationsphase digital* erfolgt auf Grundlage der in Kapitel 3 vorgestellten Codelines. Dabei wurde sich in dem Kapitel lediglich auf die Codelines von den Studien beschränkt, in denen die SuS eine leichte Strukturierung zum Beginn des Prozesses bekamen (Hilfestellung *4-Phasen analog*). In den Abbildungen 5.5, 5.6 und 5.7 sind die Codelines der Studie dargestellt, die mit LEGO Mindstorms-Robotern unter der Verwendung der Hilfestellung *Evaluationsphase digital* durchgeführt wurden.

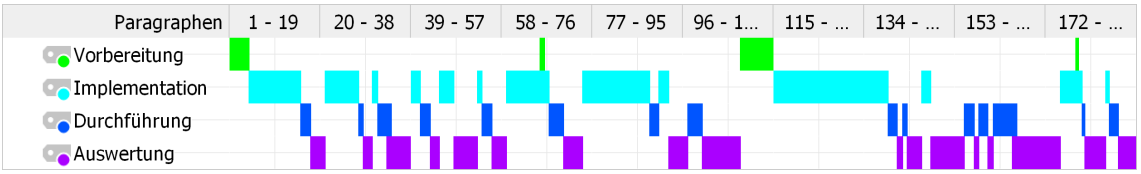


Abbildung 5.5: Codeline zur Beschreibung der Hauptphasen, Studie 3, Gruppe AB

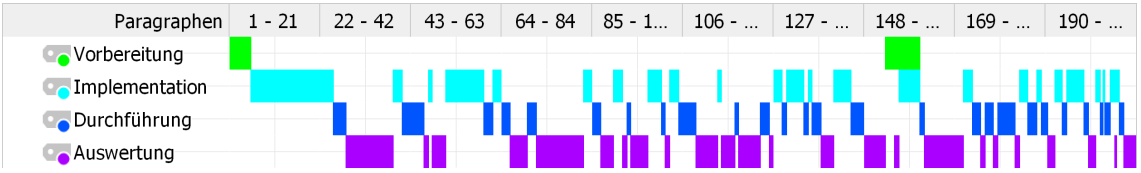


Abbildung 5.6: Codeline zur Beschreibung der Hauptphasen, Studie 3, Gruppe CD

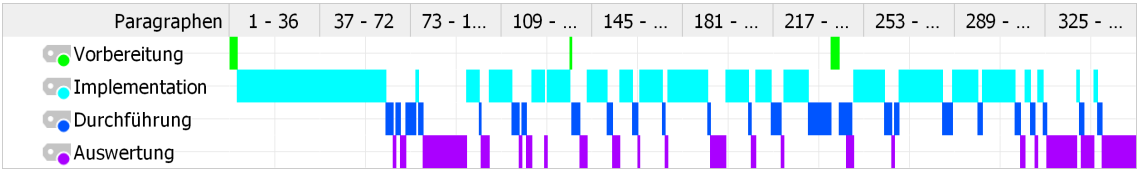


Abbildung 5.7: Codeline zur Beschreibung der Hauptphasen, Studie 3, Gruppe EF

Die Codelines zeigen einen strukturierten, iterativen Ablauf des Physical-Computing-Prozesses, in dem zumeist nur die Vorbereitungsphase ausgelassen wird. Diese ist nicht in jedem Schritt notwendig, sondern nur, wenn das prinzipielle Vorgehen geändert wird. Im Vergleich zu den Codelines der Studien 1 und 2 (vgl. Abbildung 3.20 und 3.27) scheint der Prozess (insbesondere in der ersten Hälfte) mit der Hilfestellung *Evaluationsphase digital* strukturierter abzulaufen. Dies ist ersichtlich dadurch, dass verschiedene Phasen seltener übersprungen oder Prozessdurchläufe abgebrochen werden.
In Abbildung 5.8 und 5.9 sind die Code-Relation-Darstellungen für die 3. Studie (LEGO-Roboter, Hilfestellung *Evaluationsphase digital*) angegeben. Die stärkste Relation weisen die Durchführungs- und die Auswertungsphase auf. Das bedeutet, dass diese Phasen $N = 113$ mal mit einem Zeilenabstand von maximal $N = 1$ voneinander auftreten. Sie folgen dementsprechend häufiger aufeinander als andere Phasen. Die zweitstärkste Relation weisen die Phasen der Implementation und der Durchführung auf ($N = 102$). Darauf

folgt die Relation der Implementations- und der Auswertungsphase mit $N = 89$. Die Vorbereitungsphase ist 16 mal in der Nähe der Implementationsphase zu finden und weist damit die größte Relation zu dieser auf. In der grafischen Darstellung in Abbildung 5.9 ist kein wesentlicher Unterschied in der Größe der Relation von der Vorbereitungsphase zu anderen Phasen zu erkennen. Die Ursache dafür liegt in der Methodik der Grafikerstellung von MaxQDA. Abhängig vom Maximum (hier $N = 113$) wird der Wertebereich in sieben Intervalle geteilt, denen verschieden große Quadrate zugeordnet werden. Das $N = 16$ der Vorbereitungs- und Implementationsphase liegt damit an der Intervallgrenze und befindet sich in dem gleichen Intervall wie alle darunter liegenden Zahlen.
Die Interpretation dieser Relationen in Kombination mit den Abbildungen 5.5, 5.6 und 5.7 lassen die Aussage zu, dass auf eine Durchführungsphase am wahrscheinlichsten eine Auswertungsphase folgt. Weiter ist erkennbar, dass auf eine Implementationsphase am häufigsten eine Durchführungsphase und auf die Auswertungsphase am wahrscheinlichsten eine erneute Implementationsphase folgt.
Diese Code-Relation-Darstellungen können mit der 1. und 2. Studie (vgl. Abbildung 3.10 und 3.11) mit LEGO Mindstorms-Robotern verglichen werden. In diesen Studien bekamen die SuS die Hilfestellung *4-Phasen analog* zur Verfügung gestellt, die den Physical-Computing-Prozess anhand von Leitfragen strukturierte. Die Relation der Durchführungs- und Auswertungsphase ist mit der Unterstützung *Evaluationsphase digital* am stärksten ausgeprägt. Somit kann geschlussfolgert werden, dass das Ziel erreicht wurde, dass die SuS ihre Durchführungsphase auswerteten. Im nächsten Schritt wird deshalb die Qualität der Auswertungsphase der SuS untersucht, um Aussagen über die Effektivität der Hilfestellung abzuleiten.

Codesystem	Vorbereitung	Implementation	Durchführung	Auswertung
Vorbereitung		16	7	5
Implementation	16		102	89
Durchführung	7	102		113
Auswertung	5	89	113	

Abbildung 5.8: Numerische Code-Relation-Darstellung über LEGO Mindstorms-Roboter-Studien mit Unterstützung

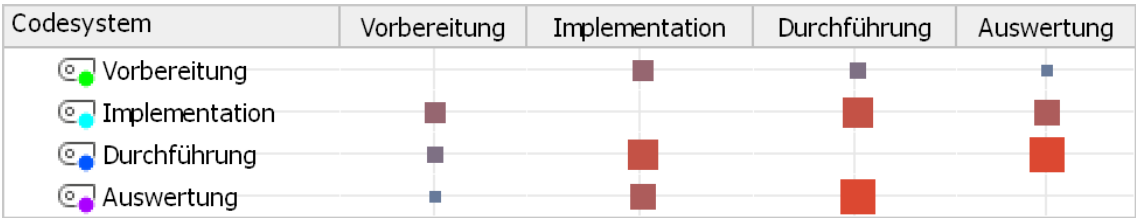


Abbildung 5.9: Graphische Code-Relation-Darstellung über LEGO Mindstorms-Roboter-Studien mit Unterstützung

5.3.2 Analyse qualitativer Daten

Die Transkripte der Studien 1 und 2 werden nun mit der 3. Studie auf inhaltliche Gemeinsamkeiten und Unterschiede während des Problemlöseprozesses verglichen. Ziel der Hilfestellung *Evaluationsphase digital* war, dass die SuS eine intensive Evaluationsphase durchführen. Als Merkmal dafür werden im Folgenden mögliche Probleme beschrieben, die die SuS als Erklärung für die Abweichung des Roboterverhaltens von dem geplanten Verhalten anführten. Dabei wird betrachtet, welche der Problemursachen (Hardware, Software und Umgebung) sie aufzeigen und ob dabei die konkrete Ursache oder nur die Auswirkungen beschrieben werden. In Tabelle 5.5 werden die Beschreibungen der Problemfelder in den Studien miteinander verglichen. In dem Bereich *Software* beschreiben die SuS beider Studien, dass sie Veränderungen an verschiedenen Variablen vornehmen sollten, die das Fahrverhalten des Roboters bestimmen. Dazu gehören die Anzahl der Radumdrehungen und die Geschwindigkeit. Bei der Verwendung von Kontrollstrukturen, wie Schaltern und Schleifen, benennen sie die Veränderung des Schwellwerts für den Ultraschallsensor und die Position der Sensorabfrage im Programm (die die Abtastrate bestimmen kann) als Probleme. Auch die Verwendung von Schleifen im Allgemeinen und die Wiederholungsanzahl werden diskutiert. Der Bereich Software weist zunächst keine Unterschiede zwischen den Gruppen auf und es wurden ausschließlich Gemeinsamkeiten gefunden.
Innerhalb des Bereichs *Hardware* beschrieben die SuS der 1. und 2. Studie, dass sie das Gefühl haben, der Sensor würde nicht reagieren. Hierbei gingen sie allerdings nicht darauf ein, worin die Ursache für dieses Problem liegen könnte. SuS der 3. Studie beschrieben hingegen, dass der Sensor nicht funktionieren würde, da er defekt sei.
Die Kabelführung (z. B. zu den einzelnen Motoren) ist ebenfalls in diesen Bereich einzuordnen. Dabei nahmen die SuS konkrete Veränderungen an der Kabelführung vor, um Probleme zu beheben. SuS der 3. Studie beschrieben ebenfalls Probleme in der Konstruktion, z. B. die Position des Ultraschallsensors, nahmen jedoch keine Veränderungen daran vor. Zusätzlich zogen die SuS in ihre Überlegungen ein, dass die beiden Motoren möglicherweise nicht die gleiche Leistung erbringen, was die Ursache für diverse weitere Probleme sein kann. Dazu gehört z. B. der Einfluss auf die programmierte und die ausgeführte Gradzahl bei Drehungen des Roboters. Bei dem Vergleich beider Gruppen sind leichte Unterschiede zu erkennen: Die SuS der 3. Studie beschrieben diverse Probleme und diese zum Teil genauer, wohingegen SuS der 1. und 2. Studie sogar Veränderungen in diesen Bereichen auf eigene Initiative vornahmen. Aufgrund der geringen Anzahl von TeilnehmerInnen, können davon jedoch keine zu verallgemeinernden Aussagen abgeleitet werden.

Tabelle 5.5: Problembeschreibungen der SuS in den Studien 1, 2 und 3

	Studie 1 und 2	Studie 3
Software	• Anzahl der Rad-Umdrehungen • Geschwindigkeit des Roboters • Entfernungseinstellungen in Schaltern • Verwendung von Schleifen • Abfragerate des Sensors im Programm	• Anzahl der Rad-Umdrehungen • Geschwindigkeit des Roboters • Entfernungseinstellungen in Schaltern • Verwendung von Schleifen • Abfragerate des Sensors im Programm
Hardware	• Sensor reagiert nicht • Position der Berührungs-sensoren ist störend • Kabelführung verändern	• Sensor funktioniert nicht • Motoren sind unterschiedlich stark • Position des Ultraschallsen-sors
Umgebung		• Fahruntergrund ist nicht glatt und gerade oder verschmutzt • Lichtverhältnisse • versehentlich Hand vor den Sensor gehalten

In Studie 1 und 2 wurde die Konstruktion des Roboters diskutiert. Die Position der Berührungssensoren wurde dabei als ein Problem beschrieben. Sie beschrieben diese Probleme mit:

Beispiel 5.3.1 *2. Studie, Gruppe DE:*

D: *Jetzt reagiert der Sensor gar nicht mehr.*

oder

Beispiel 5.3.2 *2. Studie, Gruppe DE:*

D: *Die Tastsensoren sind auch doof hier vorne.*

Die Problemursache *Umgebung* wurde ausschließlich von TeilnehmerInnen der 3. Studie beschrieben. Diese führten an, dass der Untergrund vielleicht nicht glatt genug oder ungerade sei, was Probleme beim Fahren verursachen kann. Sie überlegten ebenfalls, ob die Lichtverhältnisse die Funktion eines Ultraschallsensors beeinflussen könnten, wobei sie das anschließend selbst negierten. Neben den natürlichen Umgebungsfaktoren bezog eine Gruppe in dieser Studie auch menschliche Einflüsse als Problemursache mit ein. Lediglich eine Gruppe bemerkte, dass sie beim Starten des Programms die Hand vor dem Ultraschallsensor hatten und das dazu führte, dass der Roboter direkt in eine andere Richtung fuhr. Anhand der von den SuS beschriebenen Probleme ist die folgende Hypothese möglich: *die Vorstrukturierung von möglichen Problemursachen erhöht die Vielfalt an Problemen, die die SuS identifizieren.*

5.3.3 Diskussion und Zusammenfassung der Teilstudie zur Intensivierung der Evaluationsphase

In den Videoaufnahmen ist zu erkennen, dass die SuS trotz der Hilfestellung *Evaluationsphase digital* diese nicht für jede Auswertungsphase nutzten und erinnert werden mussten, sie auszufüllen. Die Gruppen nutzten die Hilfestellung *Evaluationsphase digital* während der Interventionsdauer zwei- bis achtmal. Der vorgenommene Vergleich deutet darauf hin, dass die SuS der 3. Studie eine intensivere Auswertungsphase durchliefen, in der sie eine größere Vielfalt an Problemursachen in Erwägung zogen. Es fällt jedoch auf, dass sie dennoch weiterhin vorwiegend Veränderungen im Bereich Software vornahmen, obwohl sie weitere Problemursachen entdeckt haben. Die durchgeführten Analysen geben Hinweise darauf, dass eine Vorstrukturierung der Evaluationsphase eine Veränderung des Physical-Computing-Prozesses hervorrufen und die Vielfalt von betrachteten Problemen positiv beeinflussen kann. Eine Kombination der prozessualen Unterstützung mit weiteren Hilfestellungen (z. B. direkten Instruktionen und Tutorensystemen) scheint notwendig zu sein. Die Tabelle 5.6 fasst die Teilstudie zusammen.

Tabelle 5.6: Zusammenfassung der Evaluation der Hilfestellung *Evaluationsphase digital*

Geräte	LEGO Roboter
Hilfestellung	*Evaluationsphase digital*als prozessuale Unterstützung
Ergebnisse	• die SuS beschrieben vielfältigere Probleme und die Problemursache Umgebung • die Lösung von Problemen verblieb auf der Systemkomponente Software • es wurden weiterhin Evaluationsphasen von den SuS übersprungen • die Hilfestellung wurde unregelmäßig ausgefüllt
Schlussfolgerung	die Prozessunterstützung bewirkt geringe Veränderungen des PhC-Prozesses bei SuS

5.4 Teilstudien Wizard-of-Oz I und II

In diesem Kapitel geht es um eine detailliertere Untersuchung der Hilfestellung *Problemtaxonomie gestuft*. Dazu wurden zwei Wizard-of-Oz-Studien in unterschiedlichen Settings durchgeführt, deren jeweilige Gruppenbeschreibung und Evaluation in den folgenden Abschnitten getrennt erfolgt. In den Studien werden die gleichen Testinstrumente genutzt, weshalb die Gemeinsamkeiten im methodischen Vorgehen zusammen beschrieben werden. Alle SuS füllten als Pretest die Testinstrumente *Test zu algorithmischen Strukturen* und *Physical-Computing-Test* aus und die gleichen Tests wurden nach der Intervention als Posttest verwendet. Die Testhefte des Physical-Computing-Tests wurden zuvor in einer Expertenbefragung von wissenschaftlichen Mitarbeitern in der Robotik (Lehrstuhl für Adaptive Systeme an der HU Berlin) auf Verständnis und Korrektheit geprüft. Nach der Durchführung des Posttests wurden 50 % der offenen Antworten gemeinsam mit einem weiteren wissenschaftlichen Mitarbeiter überprüft sowie gegebenenfalls diskutiert und die Einschätzung überarbeitet. Kriterien für die Bewertung der offenen Antworten waren:

1. fachliche Korrektheit,
2. Plausibilität des beschriebenen Szenarios,
3. Disjunktheit der einzelnen Antworten.

Trafen nicht alle drei Kriterien bei einer Antwort zu, wurde kein Punkt vergeben. Für alle Antworten war es möglich, einen Punkt zu erhalten.

5.4.1 Studiendesign der Wizard-of-Oz-Studie I

Die Studie WoZ I wurde in der Schülergesellschaft Informatik an der Humboldt-Universität zu Berlin durchgeführt. In diesem Jahr (2017) wurde das Thema *Robotik* behandelt und 15 SuS nahmen elf Wochen lang daran teil. Bei den Teilnehmenden handelt es sich um besonders begabte und interessierte Schülerinnen und Schüler, die einmal wöchentlich am Nachmittag für 2 Stunden an einem speziell für sie konzipierten Kurs teilnahmen. In diesem Semester waren SuS der Klassenstufen $7-10$ vertreten. Von den 15 TeilnehmerInnen waren sieben bereits im Vorjahr in der Schülergesellschaft und hatten durch das dort stattfindende Drohnen-Projekt Vorwissen erworben. Der Kurs wird von zwei Lehramtsstudenten, zwei wissenschaftlichen MitarbeiterInnen und zwei Studenten mit besonderem Hintergrund in der Robotik durchgeführt. Bei der Bewerbung zur Schülergesellschaft gaben alle SuS an, bereits Programmiererfahrungen zu haben, was durch einen Pretest zum Beginn überprüft wurde. In der ersten Stunde wurde das Anwenden von Kontrollstrukturen durch den Basis-Programmierkompetenz-Test (BPK-Test) erhoben. Des Weiteren wurde die Evaluation von Problemen in der Robotik mittels eines selbst konstruierten und pilotierten Tests erhoben.
Die *Interventionen* wurden wie folgt durchgeführt: Es gab jeweils fünf betreuende Personen

(Wizards), die für ein bis zwei Gruppen zuständig waren. Da von den SuS durchschnittlich sieben Gruppen anwesend waren, wurden maximal zwei von fünf Wizards zwei Gruppen gleichzeitig zugewiesen. Die anderen Wizards waren jeweils für eine Gruppe zuständig. An der 1. Interventionsstunde (mit LEGO Mindstorms-Robotern) nahmen 14 SuS und an der 2. Interventionsstunde (Nao-Roboter) 12 SuS teil. Bei der letzten Interventionsstunde (Nao-Roboter) nahmen zehn SuS teil, weshalb alle Wizards nur für eine Gruppe zuständig waren. Da die SuS ihre GruppenpartnerInnen freiwillig wählen konnten und die Gruppen zumeist nicht in der gleichen Besetzung an den Interventionsstunden teilnahmen und sich dadurch mischten, wurden die Gruppen den Wizards jeweils zufällig zugewiesen.
Jeder Wizard hatte einen ausgearbeiteten Stundenverlaufsplan, auf dem konkrete Aufgabenstellungen formuliert waren, so dass jeder Wizard der/den Gruppe/n die gleiche Aufgabe erteilte. Anschließend hielt sich der Wizard im Hintergrund und folgte einem vorgegebenem Schema der Hilfestellungen und Regeln, wann eingegriffen werden sollte (Beschreibung siehe Kapitel 5.2.2). Das verwendete Schema ist im Anhang B.3 zu finden. Während der Intervention waren die Wizards angehalten alle gegebenen Hilfestellungen zu protokollieren. Nach der Intervention von circa 75 - 90 min wurden alle Gruppen einzeln befragt. Die gestellten Fragen bezogen sich auf die Probleme der SuS beim Lösen der Aufgaben und bzgl. der gegebenen Hilfestellungen ihrer Wizards. Abschließend wurden die Wizards in einem Gruppeninterview befragt, an welchen Stellen und auf welchen Ebenen der Hilfestellungen sie Hilfe geleistet haben. Zuletzt wurden die Wizards um eine allgemeine Einschätzung bezüglich der Anwendbarkeit der Hilfestellung gebeten.

5.4.2 Durchführung der Wizard-of-Oz-Studie I

Inhaltlich wurden in den ersten drei Sitzungen durch den Einsatz von LEGO Mindstorms-Robotern (Generation EV3) zunächst Grundprinzipien der Robotik eingeführt. In der 3. Sitzung fand die 1. Intervention statt, bei der die SuS die Abtastrate des Ultraschallsensors durch verschiedene Untersuchungen erprobten. In den darauf folgenden Stunden wurden humanoide Roboter (Nao-Roboter) eingesetzt. Die SuS begannen einen Chatbot mit dem Roboter zu programmieren, der auf eine Spracheingabe Bewegungen ausführte. Nach zwei Einführungsstunden folgten aufeinander zwei Interventionsstunden (die 2. und 3. dieser Studie) mit dem Nao-Roboter. In der 2. Interventionsstunde wurde das Greifen von verschiedenen Gegenständen durch den Roboter thematisiert und der Roboter sollte als Abschluss ein Bild malen bzw. Buchstaben schreiben. In der 3. Interventionsstunde war das Ziel, dass der Roboter einen Ball durch eine Kamera erkennen und gezielt in dessen Richtung greifen kann. Als zweiten Schritt sollte der Roboter den Ball nicht nur erkennen können, wenn er vor ihm liegt, sondern auch im Raum nach dem Ball suchen, indem er den Kopf drehte und senkte. Alle erteilten Aufgaben sind in Kapitel 5.2.2 dargestellt.
Die SuS arbeiteten jeweils mit blockbasierten Programmierumgebungen. Bei den LEGO-Robotern wurde die Webanwendung *Open Roberta* genutzt, die Scratch sehr ähnlich ist

und damit den Einstieg für die SuS erleichtern sollte. Für die Nao-Roboter ist ebenfalls eine grafische Programmierumgebung namens *Choreograph* verfügbar, bei der Blöcke frei angeordnet und anschließend verbunden werden können. Die zu verwendenden Blöcke wurden zunächst von den SeminarleiterInnen vorbereitet, bis den SuS erklärt wurde, wie sie selbst Bewegungen aufnehmen und in Blöcken speichern können. Der Algorithmus zur Ballerkennung wurde gänzlich von den SeminarleiterInnen vorbereitet, so dass die SuS keine Änderungen in der generellen Ballerkennung vornehmen mussten. Die Blöcke werden mittels Python programmiert und in mehreren bereits verbundenen Blöcken zusammen gefasst. Sie wurden nach ihren Funktionen benannt, damit die SuS den Inhalt der Boxen antizipieren können. Die verwendeten Geräte und die Software sind in Kapitel 5.2 illustriert.

Die Schwierigkeit der Aufgaben im Umgang mit dem Nao-Roboter liegt besonders im Bereich der Hardware, da eine Vielzahl von Motoren als Gelenke fungieren. Die Umgebungseinflüsse nehmen ebenfalls ein größeres Ausmaß an als bei LEGO-Robotern, da mit Spracherkennung und Bildverarbeitung gearbeitet wird. In Aufgabe 1 (Ball mit beiden Armen festhalten, siehe Kapitel 5.2.2) ist eine besondere Herausforderung, dass die Motoren, die die Schultergelenke des Nao-Roboters bilden, sehr ungenau sind und nicht fest an einer genauen Gradzahl einrasten. Dadurch werden die Arme oftmals nicht nah genug zusammengeführt, um einen Ball festzuhalten. Bei der Arbeit mit Keyframes werden alle Gelenke des Roboters zu einem bestimmten Zeitpunkt aufgenommen und gespeichert. Dadurch ist es möglich, den Roboter die gleiche Position einnehmen zu lassen, ohne einzelne Motoren im Programm direkt ansteuern zu müssen. Das Aufnehmen von Keyframes stellt eine Herausforderung dar, da der Roboter nicht völlig isoliert an einer Stelle bewegt werden kann und es somit zu Ungenauigkeiten in der gesamten Körperhaltung kommt. Aus diesem Grund soll bei Aufgabe 2 nur mit dem oberen Teil des Körpers gearbeitet werden, um die ungewollten Gelenkstellungen zu minimieren und das Hinfallen des Roboters zu vermeiden. Das ist wichtig, um Schäden an dem Roboter vorzubeugen. Des Weiteren wird dadurch die Komplexität der Aufgabe reduziert, da sie aus Erfahrung der Kursleitung ansonsten nicht für die vertretenen Klassenstufen geeignet gewesen wäre.

5.4.3 Quantitative Evaluation der Wizard-of-Oz-Studie I

Der Pre- und Posttest (vgl. Tabelle D.9 und D.10) zur Evaluation von Problemen im Physical Computing zeigt eine Tendenz zur Verbesserung der SuS. Der Test erfragt zuerst die Vorerfahrungen der SuS in der Informatik allgemein und bezogen auf einige Unterkategorien (Programmierung, LEGO Mindstorms-Roboter und Physical Computing). Die SuS sollten auf einer Skala von „sehr wenig“ bis „sehr viel“ angeben, wie sie ihre Erfahrungen in diesen Bereichen einschätzen. Numerisch wird die Skala mit Werten von 1 bis 6 beschrieben, wobei die Zahl 1 der Beschreibung „sehr wenig“ und 6 „sehr viel“ entspricht.

Im Pretest gaben sie ihre Vorerfahrungen über allen Kategorien mit Mittelwerten von $2,2-4,0$ (Median $M=4$) an. Im Posttest zeigte sich eine Tendenz des Anstiegs aller Werte und es wurden Mittelwerte von $2,8-4,3$ ($M=4$) ermittelt. Das Ergebnis kann so interpretiert werden, dass die SuS über den Verlauf der zehnwöchigen Schülergesellschaft mehr Selbstvertrauen in den verschiedenen Unterkategorien erlangten, somit ihre Kenntnisse seltener in den Mittelkategorien einstuften und sich fast doppelt so oft den positiven Bereichen „5“ und „6“ zuordneten. Der Median weist jedoch keine Veränderungen im Pre- und Posttest auf. Von der Ermittlung von Effektstärken wird im Folgenden aufgrund der geringen ProbandInnen-Anzahl abgesehen.

Es wurden die Vorstellungen erhoben, in welcher Häufigkeit die Komponenten Hardware, Software und Umgebung Probleme im Bereich Robotik hervorrufen. Auf den Skalen „stimmt gar nicht“ bis „stimmt sehr“ bzw. $1-6$ gaben die SuS im Pretest an, dass die Kategorie *Software* mit $MW=4,1$ ($M=5$) am häufigsten auftritt, gefolgt von *Hardware* mit $MW=3,1$ ($M=3$) und *Umgebung* mit $MW=3,5$ ($M=4$). Im Posttest ergab sich eine leichte Veränderung, wobei sich der Stellenwert der Systemkomponenten nicht veränderte. Die SuS bewerteten den Bereich *Software* mit $MW=4,6$ ($M=5$) und damit geben sie ihm eine größere Gewichtung als beim Pretest. Der Einfluss der *Hardware* wurde mit $MW=2,8$ ($M=3$) als geringer eingeschätzt, wobei es sich hier nur um eine marginale Abweichung handelt. Im Bereich *Umgebung* zeigen sich ebenfalls nur leichte Veränderungen mit $MW=3,8$ ($M=4$). Die hier beschriebenen Tendenzen lassen sich jedoch nur aus der Betrachtung des Mittelwerts ableiten, da der Median in allen Kategorien unverändert blieb.

In der 2. Aufgabe sollten die SuS beschriebene Szenarien, z. B. *der Roboter fährt geradeaus*, der Art der Datenverarbeitung *Input*, *Processing* und *Output* zuordnen. Die letzte Spalte von Tabelle D.9 fasst den Mittelwert der richtigen Schülerantworten in den drei Kategorien zusammen. Im Bereich Input des Pretests beantworteten sie die Fragen zu 79 % richtig, für Output mit 70 % und Processing mit 88 %. Im Posttest zeigt sich eine Tendenz der Verbesserung der SuS bei der Zuordnung von Aktionen in den Bereich Output (auf 89 %). Der Bereich Input (mit 69 %) und Processing (mit 88 %) zeigen marginale bis keine Veränderungen.

In der 3. Aufgabe wurde anhand eines LEGO Mindstorms-Roboters ein konkretes Problem geschildert. Dabei sollten die SuS für jede Systemkategorie drei plausible Beispiele finden, die dieses Problem hervorgerufen haben könnten. In allen drei Bereichen haben sich die SuS verbessert. Dies ist teilweise darin zu begründen, dass sie mehr Beispiele nennen konnten, diese konkreter wurden und seltener Redundanzen auftraten. Zuvor beantworteten die SuS 40 % der Fragen richtig, wobei im Posttest ein starker Anstieg auf 62 % zu verzeichnen war. Für den Bereich *Hardware* zeigen sich nur marginale Veränderungen von 60 % auf 69 %. Die Antworten für die Kategorie *Umgebung* zeigen einen Anstieg von 50 % richtig beantworteter Fragen auf 67 %.

Die SuS zeigen in den meisten erhobenen Bereichen eine Tendenz zur Verbesserung. Sie

verbesserten die Zuordnung von Aktionen eines Roboters zu den vorgegebenen Kategorien und konnten mehr Probleme für verschiedene Problemursachen benennen. Diese Ergebnisse deuten darauf hin, dass die SuS nach der Intervention ihr Systemverständnis verbesserten und somit die Ursachen von Problemen gezielter ermitteln können.
Darüber hinaus wurde der Programmierkompetenztest eingesetzt, um die Verwendung von Kontrollstrukturen der SuS zu erheben (vgl. Anhang D). Vor der Intervention beantworteten die SuS die Programmieraufgaben zu 57 % richtig. Die Ergebnisse zeigen, dass es sowohl SuS gab, die keine Aufgabe richtig beantworteten (Minimum $Min = 0$) sowie SuS, die die höchste Punktzahl mit einem Maximum von $Max = 7$ Punkten erreichten. Aus allen Testheften wurde der $MW = 4$ und der $M = 4,5$ als durchschnittliches Testresultat ermittelt (vgl. Tabelle D.1 und D.2). Der Posttest zeigt eine leichte Verbesserung der SuS in ihrer Programmierfähigkeit. Alle SuS erreichten Punktzahlen von $Min = 1$ und $Max = 7$, und beantworteten 65 % aller Aufgaben richtig. Der $MW = 4,6$ und der $M = 5$ zeigen ebenfalls eine positive Tendenz (vgl. Tabellen D.3 und D.4). Die meisten SuS begannen die Studie mit bereits guten Programmierfähigkeiten und zeigten eine leichte Verbesserung über den Interventionszeitraum hinweg.

5.4.4 Qualitative Evaluation der Wizard-of-Oz-Studie I

Die SuS wurden nach jeder Interventionsstunde paarweise (zusammen mit dem/r GruppenpartnerIn) in einem gesonderten Raum befragt. Dieses fokussierte Interview dauerte circa 2 min und die SuS sollten kurz die erlebte Intervention reflektieren. Dabei konnte die interviewende Person ggf. nicht zutreffende Fragen überspringen. Die Interviews wurden mittels Diktiergerät aufgezeichnet. Die Aussagen der interviewenden Person wurden mit der Kennzeichnung L, für Lehrkraft, gekennzeichnet. Die Evaluation wird zunächst getrennt nach den Geräten vorgenommen, da ein Zusammenhang zwischen Gerät und Problemtaxonomie (vgl. Kapitel 4) vermutet wird. Dabei wurden den SuS folgende Fragen gestellt:

1. Habt ihr beim Lösen der Aufgaben Hilfestellungen von euren Betreuern benötigt?
2. Beschreibt bitte, falls zutreffend, welche konkreten Probleme ihr hattet!
3. Beschreibt bitte, ob ihr die gegebene Hilfestellung als nützlich empfandet! Wenn ja, was war daran hilfreich?
4. Beschreibt bitte, was ihr an der Unterstützung gegebenenfalls ändern würdet bzw. was euch besser unterstützt hätte!
5. Habt ihr noch weitere Anmerkungen?

Bezüglich der *1. Interventionsstunde* wurden sieben Gruppen (14 SuS) befragt. Sie sollten beschreiben inwieweit sie die Problemtaxonomie beim Lösen von Aufgaben mit LEGO

Mindstorms-Robotern als hilfreich empfanden. Dabei formulierten sie circa 50 Aussagen, wobei pro Person mehrere Aussagen in der gleichen codierten Kategorie möglich waren. Auf die Frage *Habt ihr beim Lösen der Aufgaben Hilfestellungen von euren Betreuern benötigt?* antworteten fünf Gruppen mit einem ausdrücklichen *ja* und zwei Gruppen gaben an, *wenig* Hilfe benötigt zu haben. Gruppen, die angaben, dass sie Hilfe benötigten, wurden ebenfalls gefragt, wie sich ihre Aussage auf die Einteilung in die aufgezeigten Kategorien bezieht. Dabei gaben zwei Gruppen an, dass sie lediglich Softwareprobleme hatten und deshalb keine weitere Einteilung benötigten. Zwei weitere Gruppen sagten, dass sie keine Hilfe bei der Einteilung benötigten, da ihnen bewusst war, in welcher Kategorie ihr Problem lag. Darauf aufbauend wurden sie gebeten: *Beschreibt bitte, falls zutreffend, welche konkreten Probleme ihr hattet!* Besonders oft wurde der Bereich *Software* beschrieben ($N = 10$) bzw. verschiedene Probleme in diesem Bereich. Als Unterkategorie der Software wurde explizit der *Umgang mit der Programmieroberfläche* als Problem genannt ($N = 6$). Die Bereiche *Hardware* ($N = 2$) und *Umgebung* ($N = 1$) wurden selten genannt. Zwei Gruppen beschrieben sogar Probleme im Bereich *mathematisch/physikalische Grundlagen.*

Beispiel 5.4.1 *1. Intervention, Gruppe EF:*

F: *Einmal hat unsere mathematische Formel nicht funktioniert, weil wir nicht wussten, ob das in cm ist und wie genau das ist und deshalb hat es nicht geklappt.*

Neben der Zuordnung zu den Hauptproblemursachen wurde bei einigen Gruppen explizit nachgefragt, ob sie Probleme hatten, wenn sich zwei Bereiche von Problemursachen überschnitten. Zwei Gruppen beschrieben diese Überschneidungen im Bereich *Software/Hardware* bereits zuvor, drei andere Gruppen gaben an, solche Überschneidungen nicht gehabt zu haben. Abhängig davon, ob die SuS zuvor Probleme beschrieben und die Hilfestellung benötigten, wurden sie gebeten *Beschreibt bitte, ob ihr die gegebene Hilfestellung als nützlich empfandet! Wenn ja, was war daran hilfreich?* oder *Beschreibt bitte, was ihr an der Unterstützung gegebenenfalls ändern würdet bzw. was euch besser unterstützt hätte!* Die Mehrheit beschrieb die Hilfestellung als allgemein hilfreich ($N = 4$), ohne es genauer zu erklären. Zwei Gruppen wiesen darauf hin, dass sie insbesondere die Unterkategorien innerhalb der Software benötigten und die Hilfestellung nützlich war, um diese zu identifizieren. Eine Gruppe gab an, dass sie die Hilfestellung für diese Aufgaben kaum brauchten, sie jedoch bei komplexeren Aufgaben für sinnvoll erachten. Als Verbesserungsmöglichkeit für die Taxonomie nannte eine Gruppe, dass sie Probleme damit hatten, herauszufinden wann ein neuer Fehler im gleichen Bereich vorlag. Dadurch, dass lediglich auf die Kategorie Software verwiesen wurde, konnten sie sich nicht sicher sein, ob es sich um den gleichen oder einen neuen Fehler handelte. Auf die Frage, ob sie *noch weitere Anmerkungen* haben, erwähnten zwei Gruppen, dass die Stunde in dieser Form gut war und Spaß gemacht habe. Drei Gruppen wurden gefragt, ob sie es für *sinnvoll halten mit dem Feedback in dieser Form in den nächsten Stunden weiter zu machen.* Zwei Gruppen

bejahten diese Aussage, eine weitere wies darauf hin, dass sie es abhängig von den konkreten Aufgaben für nützlich halten.
Zusammenfassend kann gesagt werden, dass die SuS die Hilfestellung gut annahmen, wobei sich ihr Einsatzbereich momentan noch vorrangig auf den Bereich Software bezog. Der Test zur Einsatzfähigkeit der gesamten Taxonomie erfordert komplexere Aufgaben und ggf. verschiedene Soft- und Hardwarekomponenten sowie wechselnde Umweltbedingungen. Der allgemeine Nutzen wurde jedoch nicht in Frage gestellt.
Aufbauend auf die 1. Interventionsstunde mit LEGO Mindstorms-Robotern wurden die 2. und 3. Intervention mit Nao-Robotern durchgeführt und ein gemeinsames Codesystem erstellt (Tabelle 5.7). Die Interviewfragen setzten den Fokus auf konkrete, aufgetretene Probleme (Frage 2) und darauf, welches Feedback hilfreich war (Frage 3), um die Hilfestellung *Problemtaxonomie gestuft* zu evaluieren. Die Antworten auf die erste Frage, ob die SuS Hilfe von ihren Betreuern brauchten, beziehen sich nach wie vor auf die Erfahrungen in der LEGO Mindstorms-Intervention. Innerhalb der befragten Gruppen wurde achtmal beschrieben, dass sie Hilfe benötigten, ohne dass diese näher ausgeführt wurde. Eine Gruppe nahm *viel Hilfe* in Anspruch, drei weitere Gruppen benötigten lediglich *wenig Hilfe* durch ihre Betreuer. Aufgrund der geringen Resonanz nach der ersten Intervention, wurde die *Nützlichkeit der Einteilung von Problemen in Bereiche* nicht erneut thematisiert. Zwei Gruppen gaben an, *ausschließlich im Bereich Software Probleme gehabt* zu haben, drei weiteren Gruppen war *der Bereich des Problems bereits bekannt.* Die Antworten in der Kategorie *aufgetretene Probleme* veränderten sich hingegen stark in den weiteren Interventionen. Die angegebenen *Überlappungen von Problembereichen* nahmen zu ($N = 17$). *Überschneidungen im Allgemeinen* wurden zweimal beschrieben. Mit der gleichen Häufigkeit wurden Überschneidungen des Bereichs *Hardware/Umgebung* angegeben.

Beispiel 5.4.2 *2. Intervention, Gruppe KM:*

K: *Da wir ja hauptsächlich Umweltprobleme hatten. Wie gesagt, irgendwie ich bleib* [*der Roboter*] *an einer Kiste hängen, ich bleib an mir selbst hängen, ich* [...] *bewege meine Arme gerade nicht dahin, wo ich hingehen soll. Also Hardware-Problem. So vom Prinzip her.*

Probleme, die im Bereich *Hardware/Software* lagen, wurden sechsmal explizit benannt. Überschneidungen des *Gesamtsystems* sind viermal genannt worden.

Beispiel 5.4.3 *2. Intervention, Gruppe AB:*

L: *Ok, ihr habt eine Weile gebraucht, bis ihr raus hattet, in welcher Höhe die Hand gehalten werden muss und wo der Stift genau malt.*

B: *Das (war) gar nicht so, bloß haben wir das bewegt und dann bewegte sich auch die Höhe so ein bisschen und darauf haben wir nicht geachtet.*

L: *Ahh.*

A: *Wir konnten nicht eine feste Position bestimmen, das hat sich immer ein bisschen geändert und hat dann auch nicht genau auf das Blatt gezeichnet.*

L: *Ah, und was denkt ihr, was die Ursache davon ist? Was war daran so schwierig?*

B: *Wahrscheinlich weil, wenn wir das so bewegen, ist da ja schon Dynamik drin und das man dann halt einfach das in der Luft macht.*

Angaben dazu, dass *keine Überlappungen auftraten*, wurden dreimal getätigt und beziehen sich im Wesentlichen auf die 1. Intervention. Die genannten Problemursachen waren *Hardware* mit $N = 8$, *Umgebung* mit $N = 7$ und *Software* mit $N = 19$. *Software* wurde in den Beschreibungen näher differenziert als *Bedienung der Programmieroberfläche* mit $N = 11$. Es zeichnete sich in diesem Bereich eine inhaltliche Verschiebung vom Umgang mit den Programmierblöcken hin zu verschiedenen Systemanwendungen ab, z. B. das Erstellen von Dialogen.

Beispiel 5.4.4 *2. Intervention, Gruppe KM:*

K: [...] *als wir dann halt einen neuen Dialog eingespeichert hatten und aber in eine andere Datei reingeschrieben haben, wussten wir halt nicht, dass wir dann in eine andere Datei reinschreiben müssen. Da mussten wir dann halt kurz fragen. [Die Regeln für den Dialog wurden in der falschen Datei gespeichert.]*

Die SuS beschrieben außerdem das Problem, dass sie bei den vorgegebenen Programmblöcken nicht sicher wussten, welche Aktionen diese auslösen sollen, da sie teilweise missverständlich benannt waren. Es fällt jedoch auf, dass in den zwei weiteren Interventionen (2. und 3.) lediglich vier von zehn Aussagen den Bereich *Bedienung der Programmieroberfläche* adressieren. Trotz der neuen und komplexen Programmieroberfläche sind das wenige Aussagen in diesem Bereich, was auf eine Verschiebung der Probleme in andere Bereiche hindeutet. Der Bereich *mathematische/physikalische Grundlagen* wurde zweimal genannt und veränderte sich damit nicht nach der 1. Intervention. Auf die Frage, *wie hilfreich das Feedback war*, tätigten die SuS vielfältige Angaben. Eine Aussage thematisierte den speziellen Fall, dass Programmblöcke vom Wizard erklärt wurden, was diese Gruppe als besonders hilfreich empfand. Andere Gruppen beschrieben, dass sie durch die Hilfestellung Probleme überhaupt ($N = 4$) oder besser ($N = 2$) lösen konnten. Des Weiteren berichteten einige Gruppen, dass die Hilfestellungen zu einem besseren Verständnis beitrugen. Zum Beispiel wurden die getätigten Fehler besser verstanden ($N = 2$) und das Verständnis beugte vor, diese Fehler erneut zu begehen ($N = 3$). Auch die Übertragbarkeit von Lösungen auf andere Probleme ($N = 1$) sowie das Verständnis von Problemlösestrategien im Allgemeinen ($N = 1$) wurde beschrieben.

Beispiel 5.4.5 *2. Intervention, Gruppe KM:*

M: *Auch hilfreich ist, wenn man halt zugucken kann, wie jemand anderes den Fehler löst. Also da gucken, da gucken, da gucken. Und beim 5. Mal gucken, findet man es dann. Also halt diese Reihenfolge, wie man dann quasi guckt.*

L: *Ok.*

M: *Das finde ich sehr hilfreich. Weil dann hat man selbst eben Plan, wie man beim nächsten Mal Fehler suchen, quasi gucken kann.*

L: *Also die Strategie vom Fehlersuchen meinst du?*

M: *Also einfach vom Prinzip beobachten, wie der Fehlersuchende das tut. Also erstmal am wahrscheinlichsten Punkt nachgucken und dann am nächst wahrscheinlichsten und so weiter. Und halt die Punkte, wo man die Fehler findet, raus suchen.*

Es wurde als hilfreich beschrieben, dass die Zuordnung der Probleme in Unterkategorien vorgenommen wurde ($N = 2$) und dass die Taxonomie eine direkte Zuordnung ermöglicht ($N = 1$). Acht Gruppen beschrieben das Feedback allgemein als hilfreich, ohne es genauer zu spezifizieren. Aus der Beschreibung der SuS wird deutlich, dass das gestufte Feedback von ihnen wahrgenommen und genutzt wurde. Es sollte in weiteren Studien geprüft werden, ob die folgende Hypothese bestätigt werden kann: *Durch gestuftes Feedback verbessern SuS ihre Problemlösekompetenzen.*
Der Bereich *Feedback war nicht hilfreich* wurde in den weiteren Interventionen nicht mehr explizit erfragt, da inzwischen alle Gruppen beschrieben, dass sie eine Form der Hilfestellung benötigten. Eine Gruppe erwähnte im Interview, dass das Feedback nicht detailliert genug war, um den Fehler beheben zu können, was mit den Stufen des Feedbacks zu erklären ist. Darüber hinaus wurden Programmierblöcke missverständlich benannt, so dass sie eine andere Funktion suggerierten als sie erfüllten ($N = 1$). Es gab keine weiteren Anmerkungen von den SuS und Veränderungsvorschläge für das weitere Vorgehen wurden nicht mehr explizit erfragt.
Die Vielfalt des erhaltenen Feedbacks nahm über den Zeitraum der drei Interventionen ebenfalls zu. Prinzipiell wurde beschrieben, dass das *Feedback hilfreich war*. Die genaue Darstellung der verschiedenen Aussagen ist in Tabelle 5.7 enthalten.

Tabelle 5.7: Codesystem der Aussagen von SuS in der Wizard-of-Oz-Studie I

Code-Bezeichnung	**Anzahl**
Hilfe von Betreuer wurde benötigt	
Allgemeine Beschreibung, dass Hilfe benötigt wurde	8
Viel Hilfe benötigt	1
Wenig Hilfe benötigt	3

Hilfe bei der Einteilung in Bereiche wurde nicht benötigt	
Ausschließlich Software-Probleme aufgetreten	2
Problembereiche bereits bekannt	3
Aufgetretene Probleme	
Überlappung von Bereichen	
Allgemeine Beschreibung von auftretenden Überlappungen	2
Hardware/Umgebung	2
Hardware/Software	6
Gesamtsystem	4
Keine Überlappungen aufgetreten	3
Mathematik/Physik	2
Umgebung	7
Hardware	8
Software	
Allgemeine Beschreibung von Softwareproblemen	19
Bedienung der Programmieroberfläche	11
Feedback war hilfreich	
Erklärung von Programmblöcken wurde gegeben	1
Fehler wurden dadurch verstanden	2
Problemlösestrategien wurden verstanden	1
Lösung auf andere Probleme dadurch übertragbar	1
Problem konnte gelöst werden	4
Zuordnung der Probleme in Unterkategorien	2
Wiederholung von Fehlern vermieden	3
Für umfangreiche und schwere Aufgaben anwendbar	1
Problem wurde leichter gelöst	2
Direkte Einordnung des Problems möglich	1
Unterkategorien innerhalb von Software identifizierbar	2
Allgemein hilfreich	8
Feedback war nicht hilfreich	
Benennung der Programmblöcke missverständlich	1
Hinweise nicht detailliert genug	1
Verschiedene Probleme in gleicher Kategorie nicht unterscheidbar	1
Form des Feedbacks für Folgestunden hilfreich	
Teilweise, abhängig von der Aufgabe	1
Hilfe war nützlich	2

Anmerkungen	
Stunde war gut	1
Stunde hat Spaß bereitet	1

Die Betreuer wurden nach der Durchführung der Interventionsstunden und Verabschiedung der SuS in einem Gruppeninterview befragt. Dabei handelte es sich ebenfalls um ein fokussiertes Interview mit einer Dauer von circa 15 min, welches aufgezeichnet wurde. Die gestellten Fragen lauten wie folgt:

1. Beschreibt bitte, was bei der Verwendung der Hilfestellung gut funktioniert hat und was vielleicht nicht funktioniert hat!
2. Brauchten die SuS Hilfe? Wenn ja, wo? Wenn nein, warum nicht?
3. Was sollte zum nächsten Mal konkret verändert bzw. verbessert werden?
4. Haltet ihr die Taxonomie für geeignet? Warum haltet ihr sie für geeignet bzw. warum nicht?
5. Habt ihr noch weitere Anmerkungen?

In dem Interview tätigten die vier Betreuer circa 30 Aussagen zu den oben genannten Fragen. Für die Wizards war es ebenfalls möglich, mehrere Aussagen in der gleichen Kategorie zu tätigen. Zunächst werden die Erläuterungen zu *Beschreibt bitte was bei der Verwendung der Hilfestellung gut funktioniert hat und was vielleicht nicht funktioniert hat!* beschrieben. Es wurden sechs Aussagen getroffen, was bei der Verwendung der Hilfestellung gut funktionierte. Zwei Wizards gaben dabei unterstützende Hinweise, die sie im Anschluss als sehr positiv bewerteten. Dazu gehört, dass sich die SuS in den Roboter *hineinversetzen* sollten und sich an die gleiche Stelle wie der Roboter stellen sollten, um mögliche Probleme zu entdecken. Die anderen Aussagen bezogen sich konkret auf die Hilfestellung. Es wurde angemerkt, dass die vorgegebenen Stufen gut funktionierten. Ebenfalls wurde beschrieben, dass viele Hilfestellungen im Bereich *Software* gegeben werden mussten und diese ebenfalls gut funktionierten ($N = 1$). Ein Wizard beschrieb, dass er eine informatik-affine Gruppe hatte und daher zumeist nur einen Hinweis auf einer Ebene geben musste und die SuS im Anschluss selbstständig die Lösung fanden. Als positiver Aspekt wurde genannt, dass durch die Taxonomie aufgezeigt wird, dass es neben der Software andere Problemursachen gibt. Die SuS neigten dazu, nur in der Software nach Fehlern zu suchen ($N = 1$). Bei der Beschreibung, was *nicht gut funktionierte*, wurden acht Aussagen formuliert. Bezogen auf die Taxonomie wurde genannt, dass die Kategorie *Software* nicht aussagekräftig genug sei, da die möglichen Fehler darin sehr vielfältig sein können ($N = 2$). Zwei Wizards beschrieben, dass sie Verständnisprobleme mit den verschiedenen Ebenen hatten. Ein Betreuer musste oft direkt in die 4. Ebene übergehen, da den SuS die anderen Ebenen nicht ausreichend viele Erklärungen geliefert hätten.

Beispiel 5.4.6 *1. Intervention, Wizard 3:*

W3: *Also ich hab es noch gehabt, dass wenn sie mir jetzt gesagt haben: ja, wie wäre es denn jetzt, wenn wir das ändern? Dann habe ich gesagt, ja das ist eine gute Idee oder nee mach das mal lieber anders. Weil einfach, wenn sie mir jetzt so eine direkte Frage stellen, und ich ihnen dann sage, nee ich kann das jetzt nur nach diesem Muster beantworten. Dann ging das halt nicht wirklich. Oder wenn halt absehbar war, dass sie nicht voran kommen und man weiß, ich muss das kurz klären, damit es irgendwie weitergeht, dann bin ich auch mal direkt ins Detail gesprungen. Einfach um eine Hürde zu nehmen, weil sie sich sonst endlos lang aufgehalten hätten.*

L: *Ah, quasi bist du dann direkt auf 4 gesprungen?*

W3: *Ja, weil manche Sachen uns einfach zu lange aufgehalten hätten. Wegen fehlender Vorkenntnisse und so.*

Ein weiterer Wizard beschrieb, dass er die 2. Ebene nicht benötigte, da es sich immer um die eigene Software der SuS handelte. Es wurde angesprochen, dass das Verändern der Hardware nicht einfach möglich war, da keine weiteren Bauteile zur Verfügung standen. Des Weiteren wurden zwei allgemeine Probleme angesprochen, die es erschwerten Feedback zu geben. Zum einen machten die SuS viele Fehler gleichzeitig, was es erschwerte ihnen in nur einem Bereich Feedback zu geben ($N = 2$). Andererseits fiel es einem Betreuer schwer die SuS zu der Lösung eines Problems zu führen, da sich die Lösungswege und Probleme durch das Handeln der Gruppen schnell ändern können.

Beispiel 5.4.7 *1. Intervention, Wizard 4:*

W4: *SuS sind zu schnell wieder abgelenkt, bauen schnell das Programm um, wodurch sich die Lösung für das Problem schnell ändern kann.*

Die Beschreibung des Wizards deutet auf einen zügigen Prozessverlauf und ein teilweise unstrukturiertes Arbeiten der SuS. Es gilt zu klären, ob eine stärkere Strukturierung des Prozesses unterstützend wirken kann, um gezieltes Feedback zu erteilen. Eine mögliche Hypothese ist: *Durch eine Vorstrukturierung des Physical-Computing-Prozesses für SuS können Hilfestellungen effektiver von Lehrkräften erteilt werden.*
Auf die Frage *Brauchten die SuS Hilfe? Wenn ja wo? Wenn nein, warum nicht?* antwortete ein Wizard, dass die SuS lediglich im Bereich *Software* Hilfe benötigten. Von Überschneidungen bei den Problemursachen wurde ebenfalls berichtet ($N = 2$). Dabei handelte es sich z. B. um die Veränderung der *Umgebung*, da die Software darauf nicht genügend abgestimmt war.

Beispiel 5.4.8 *1. Intervention, Wizard 1:*

W1: *Bei uns war tatsächlich der Fall, wo ich gesagt habe, ob Software oder Umgebung. Das war jetzt die Kabelkiste [..] (Kabel an Kisten hängen geblieben). Und dann haben*

sie tatsächlich angefangen das Labyrinth zu Begradigen oder die Kisten zu verschieben, also die Umgebung zu verändern.

L: *Als an der Hardware selber was zu machen?*

W1: *Als an der Software oder Hardware was zu verändern.*

Als weitere Überschneidung von Problemursachen beschrieb ein Wizard, dass Probleme vorlagen, die in der Software oder der Hardware lagen. Zur detaillierten kritischen Reflexion wurde gefragt: *Was sollte zum nächsten Mal konkret verändert bzw. verbessert werden?* In diesem Bereich wurden vier Aussagen getroffen: Zum einen sollte die Taxonomie einen höheren Detailgrad abdecken, um im Bereich Software ein ausdifferenziertes Feedback geben zu können. Die Aufgabenstellungen sollten komplexer werden, so dass es möglich ist, den SuS zu helfen und sich auftretende Probleme über die gesamte Taxonomie erstrecken ($N = 2$). Ein Wizard schlug vor, den Roboter von den SuS teilweise selbst zusammenbauen zu lassen. Das soll wahrscheinlich einen Impuls setzen, Veränderungen an der Hardware vorzunehmen, da sich die SuS mit der Bauweise des Roboters vertraut gemacht haben. Abschließend wurden sie mit *Haltet ihr die Taxonomie für geeignet? Warum haltet ihr sie für geeignet bzw. warum nicht?* zur Taxonomie im Allgemeinen befragt. Von den getroffenen acht Aussagen bezogen sich sieben darauf, dass bedingt durch die Aufgabenstellungen die Fehler vor allem in einem Bereich auftraten. Aus diesem Grund könne man noch keine Aussage zur Taxonomie treffen. Ein Wizard beschrieb, dass er die Taxonomie trotzdem für geeignet hält, da sie den SuS half ihre Probleme den Problemursachen zuzuordnen.

Beispiel 5.4.9 *1. Intervention, Wizard 1:*

W1: *Ich finde es trotzdem sinnvoll. Weil ab und zu haben sie halt gesagt „an der Software“ (und) realisiert, wo die Fehler lagen.*

Darüber hinaus beschrieben die Wizards, dass es ihnen schwer fiel den SuS nicht zu viele Informationen zu geben und die Reihenfolge der verschiedenen Ebenen einzuhalten. Des Weiteren war es schwierig, wie und ob sie auf Nebenfragen reagieren sollten. Es ist davon auszugehen, dass die SuS aufgrund der körperlichen Präsenz der Wizards und der freien Lernsituation weniger darauf achteten, ob es sich um problembezogene Fragen handelte, die sie den Wizards stellen durften.
Die Wizards wurden nach der 2. und 3. Intervention erneut in einem Gruppeninterview befragt. In diesen beiden Interviews formulierten die Wizards circa 40 Aussagen. Der Interviewleitfaden entspricht dem der 1. Intervention, wobei sich der Fokus auf den praktischen Einsatz der Taxonomie verschob. Auf die Frage *was gut funktionierte* wurden drei Aussagen darüber getroffen, dass die SuS nach dem Feedback schnell verstanden haben, was zu tun ist. Ein Wizard sagte, dass nach dem Lösen eines Problems der Transfer des Wissens auf andere Probleme stattgefunden hätte. Bezogen auf die Taxonomie beschrieb ein Wizard, dass die Anwendung der einzelnen Ebenen sehr gut funktionierte. Dabei wies er darauf hin, dass seine Schülergruppe Vorwissen mitbrachte. Mehrere Aussagen unterstützen

diese Aussagen, indem sie beschrieben, dass sich die Taxonomie besonders gut zur Binnendifferenzierung eigne, da die Niveaustufen individuell angepasst werden können. Eine dieser Aussagen wird von folgendem Zitat widergespiegelt.

Beispiel 5.4.10 *2. Intervention, Wizard 3:*

W3: *Also, was mir halt aufgefallen ist, also jetzt nicht an einer Gruppe, sondern wenn man jetzt mal alle Gruppen zusammen nimmt, was wir jetzt gerade gesagt haben. Dann merkt man ja schon, dass einige ein bisschen mehr Hilfe gebraucht haben und andere ein bisschen weniger. Und im Hinblick darauf ist das natürlich total gut, weil man den Leuten halt nicht zu viel erzählt. Die haben halt immer noch das Gefühl sich den Rest selber erarbeitet zu haben. Wie bei mir in der Gruppe, die haben sich das meiste selber erarbeitet, aber wenn die einfach nicht drauf kommen, dann ist halt gut immer ein bisschen tiefer zu gehen, somit kann man halt besonders, wenn man die Leute nicht kennt, auf die Bedürfnisse von allen ganz gut eingehen. Das ist jetzt so das, was ich von dem, was gesagt wurde, rausgehört habe.*

Zweimal wurde beschrieben, dass bereits der Hinweis auf die 1. Ebene ausreichte. Dies wurde für die Problemursachen Software und Hardware beschrieben. Alle weiteren Aussagen in diesem Bereich beschreiben Unterkategorien, die zur 1. Intervention getroffen wurden. Diese Aussagen beziehen sich insbesondere darauf, dass lediglich Probleme im Bereich Software auftraten.
In den Interviews wurde erneut thematisiert, was aus der Sicht der Wizards nicht oder ungenügend funktionierte. Eine Gruppe beschrieb dabei weitere Softwareprobleme, da ihnen die Software Choregraphe sowie die Firmware des Nao-Roboters abstürzte. Neben den SuS wiesen die Wizards ebenfalls auf die missverständliche Benennung von Boxen hin. Diese Boxen waren vorprogrammiert und wurden benötigt, um die Aufgabe zu lösen. Das Vorwissen der SuS wurde von einem Wizard als Problem beschrieben, da die einzelnen Stunden aufeinander aufbauten. Waren die SuS in der 2. Interventionsstunde nicht anwesend, so hatten sie Schwierigkeiten in der 3. Intervention zu folgen. Am häufigsten wurden Probleme beim Umgang mit den Stufen der Taxonomie beschrieben ($N = 6$). Es wurde genannt, dass es allgemein schwierig sei, die Stufen von oben bis unten durchzugehen ($N = 2$), die 2. Stufe kaum benötigt wurde ($N = 1$) oder direkt auf die 4. Stufe gesprungen werden musste ($N = 3$). In dem folgenden Beispiel beschreibt ein Wizard, in welchen Situationen der direkte Sprung auf die 4. Stufe nötig war.

Beispiel 5.4.11 *2. Intervention, Wizards 3:*

W3: *Vielleicht ein bisschen in die andere Richtung aber gleiche Folge daraus. Ich hatte zweimal, dreimal das Problem, dass ich ihnen eine Hilfestellung geben wollte, aber ich diese Abstufung einfach nicht machen konnte. Zum Beispiel haben die den Stift zu weit unten gehalten und natürlich funktioniert das, aber dann malen die halt den*

Roboter an. Na dann kann ich ja schlecht sagen, na probiert das erstmal, weil dann ist der Roboter angemalt. Da muss ich ihnen halt gleich sagen. Also wenn ich sage, ihr habt ein Hardware-Problem, da kommen die halt nie drauf, dass sie jetzt den Stift höher halten sollen. Weil für die ist das ja in dem Moment kein Problem, deshalb muss ich denen halt schnell sagen, dass sie den Stift weiter oben anfassen sollen.

Feedback, das sich direkt auf die Taxonomie bezog, wurde durch zwei weitere Aussagen gegeben. Ein Wizard erklärte zweimal, dass die Taxonomie nicht den Umgang mit Tools abdecke, z. B. Dialoge in Choregraphe. Eine weitere Idee war, dass eine Kategorie implementiert werden sollte, die ein Feedback dazu ermöglicht, wenn die SuS bei der Lösung des Problems in eine falsche Richtung dachten. Weitere Probleme waren, dass die SuS zum Teil die Aufgaben nicht richtig verstanden hatten. Dieses Problem steht jedoch in keinem direkten Zusammenhang mit der Taxonomie. Das unsystematische Vorgehen der SuS wurde als schwierig beschrieben, denn die SuS veränderten zügig ihr Programm und damit veränderte sich die Lösung des Problems schnell und der Wizard musste darauf reagieren. Die eingeschränkten Möglichkeiten, die Hardware zu verändern, wurden von einem Wizard beschrieben, da er es aus diesem Grund für schwierig erachtete, ein Feedback zur Hardware zu erteilen. Die Unterkategorien *viele Fehler gleichzeitig* und *Software als Aussage half nicht* wurden in der 2. und 3. Intervention nicht mehr thematisiert.
Die Frage nach den Bereichen, *wo die SuS Hilfestellungen benötigten*, wurde nicht mehr explizit erfragt. Es wurden von den Wizards trotzdem die Hauptproblemursachen Software, Hardware und Umgebung beschrieben. Das deutet darauf hin, dass die Breite der Taxonomie genutzt wurde. Zwei Wizards beschrieben, dass sie den SuS insbesondere helfen mussten einen geeigneten Lösungsweg zu finden. Im folgenden Beispiel beschreibt ein Wizard, dass den SuS nicht zwangsläufig das Wissen über einen geeigneten Lösungsweg fehlte, sondern sie sich für eine möglicherweise schwierigere Lösung entschieden.

Beispiel 5.4.12 *2. Intervention, Wizard 1:*

W1: *Was bei mir* [...] *war, die haben das mit der Rest-Position gemacht, weil ich das auch nochmal gesagt hatte. Wenn die dann einfach sagen, die End-Position ersetzen und dann stößt er mit seinen Händen oder Beinen gegen die Box. Und ihre Lösung für das Problem war halt für den Zeitpunkt: Wir verschieben die Box* [*hin und her während der Aktion*]. [*Das ist*] *so ein bisschen seltsam, weil die haben halt einen Roboter, den sie programmieren können, warum benutzen die nicht den und da ist keiner drauf gekommen* [...]. *Und da hab ich dann gesagt: „Na, ihr seht ja, wo das Problem ist, ihr stoßt da dran. Deshalb passiert das nicht, was ihr euch gedacht habt.“ Und dann sind sie nicht, ohne dass man explizit gesagt hat, was sie machen sollen, drauf gekommen, dass sie einfach die Arm-Position verändern. Denen war völlig klar, um das zu machen, müssen sie einfach die Arm-Position dazwischen machen. Das ist denen schon klar, aber die sehen das halt einfach nicht so als den Lösungsweg, den sie gehen wollen.*

Weitere Hilfestellungen waren nötig, in dem die SuS überredet wurden, ihre Programme zu testen, wenn sie sich nicht sicher waren, ob es funktionieren kann.
Die Aussagen zu *Verbesserungsvorschlägen für die Taxonomie* verlagerten sich nach der 2. und 3. Intervention stark. Es wurde hier erneut angemerkt, dass die Bezeichnung der Boxen verbessert werden sollte. Ein Wizard war der Meinung, dass es nichts zu verbessern gäbe, da die Stunde gut verlaufen ist. Ein Verbesserungsvorschlag thematisierte die Möglichkeit Meta-Vorlagen für die SuS zu geben ($N = 3$). Somit könnte verhindert werden, dass die SuS einen zu komplizierten Ansatz wählen. Ein konkreter Vorschlag diesbezüglich war die Vorgabe, dass der Nao-Roboter immer zu Beginn und am Ende des Programms die Rest-Position (vgl. Hocke) einnehmen muss. In dieser steht er stabil und das Aufnehmen von Keyframes wird erleichtert.
Die Frage danach, ob sie die *Taxonomie für geeignet halten*, wurde ebenfalls sehr positiv beantwortet. Ein Wizard sagte, dass er die Taxonomie als intuitives Vorgehen einschätzt und Probleme auf eine ähnliche Weise analysiert. Das drückte der Wizard wie folgt aus.

Beispiel 5.4.13 *2. Intervention, Wizard 1:*

W1: *Ich glaube letztlich, wenn ich meine eigene Hilfestellung jetzt mal unabhängig von denen hier reflektiere, ist das tatsächlich so ähnlich. Dass ich erst überlege und dann nach und nach den Fehler, den die gemacht haben, versuche einzuschätzen. Erstmal so allgemeine Tipps zu geben und dann immer genauer zu werden. Das ist ja eigentlich genau das: So Gewichtung des Tipps.* [...] *Nur, dass ich eben nicht daran denke: Oh ich hab ja diesen Zettel bekommen, ich muss erstmal diese Stufe machen. Sondern je nachdem, wenn es halt welche sind, wo ich sehe: Oh, die haben es jetzt selber schon 5 Minuten versucht, dann würde ich halt gleich bei Stufe 4 einsteigen, weil ich weiß, die haben das alles schon probiert, was in Stufe 1 bereitgestellt ist.* [...] *So, das ist mir halt sehr oft aufgefallen. Aber vom Prinzip her, ist das, glaube ich, eine recht natürliche Abstufung.*

Es herrschte Einigkeit darüber, dass das Schema der Taxonomie gut angewandt werden kann ($N = 7$). Lediglich ein Wizard bezeichnete die Taxonomie in der 2. Intervention als zu starr. Dabei bemängelte er vor allem das stufenweise Vorgehen und die Tatsache nicht direkt sagen zu können, an welcher Stelle ein Problem vorliegt. Aussagen über die Konzentration auf einen Bereich von Problemursachen (wie zuvor auf den Bereich Software) wurden in den Interventionen mit dem Nao-Roboter nicht wiederholt. Aufgrund der Beschreibungen wird die folgende Hypothese abgeleitet: *Die Problemtaxonomie ist ein geeignetes Mittel für Lehrkräfte, um SuS Hilfestellungen im Physical-Computing-Unterricht zu geben.*
Die Aussagen der Wizards in den Interviews der 2. und 3. Intervention sind in der Tabelle 5.8 dargestellt.

Tabelle 5.8: Codesystem der Wizard-Aussagen in der Wizard-of-Oz-Studie I

Code-Bezeichnung	**Anzahl**
Es funktionierte gut	
SuS haben verstanden, was zu machen war	3
Transfer auf andere Probleme	1
Anwendung der Feedback-Ebenen	1
Binnendifferenzierung durch Ebenen	3
Hinweis auf Ebene 1 ausreichend	2
Stufen des Feedbacks	1
Andere Bereiche als Software aufzeigen	1
Allgemeine Hinweise	2
Es funktionierte ungenügend	
Programmierumgebung/Roboter abgestürzt	2
Missverständliche Benennung von Boxen	2
Grundlagen der SuS unzureichend	1
Stufen konnten nicht durchgegangen werden	6
Taxonomie deckt Umgang mit Tools nicht ab	2
Unterkategorie für „in falsche Richtung denken“ fehlt	1
Aufgabenstellung verstehen	1
Unsystematisches Vorgehen der SuS ist schwierig	1
Hardware ändern nicht möglich	1
Viele Fehler gleichzeitig	2
Software als Aussage half nicht	2
Wo benötigten die SuS Hilfestellungen	
Hardwareprobleme	2
Umgebungsprobleme	1
Ausschließlich in Software	1
Überschneidungen der Problemursachen	3
Geeigneten Lösungsweg finden	2
Mut zusprechen, es einfach auszuprobieren	1
Was sollte verbessert werden	
Lediglich die Boxenbezeichnung	1
Nichts, die Stunde lief gut	1
Meta-Vorlage geben	3
Detailgrad der Taxonomie erhöhen	1
Ausgewogene Problemursachen in Aufgaben schaffen	2
Roboter teilweise selbst bauen lassen	1
Ist die Taxonomie geeignet	
Taxonomie eignet sich gut	9

Taxonomie ist zu starr	1
Konzentration der Fehler durch die Aufgaben	7

Zusammenfassend fällt auf, dass sich die Aussagen nach der 1. Intervention stark von denen der 2. und 3. Intervention unterscheiden. Die in dieser Taxonomie enthaltenen Problembereiche wurden durch die Auswahl an Aufgaben und Geräten abgedeckt. Aufgrund dessen wurde die Taxonomie und das dementsprechende Feedback als hilfreiches und intuitives Vorgehen der Problemlösung beschrieben.

5.4.5 Diskussion und Zusammenfassung der Wizard-of-Oz-Studie I

Insbesondere in der 1. Intervention wurde die Breite der Taxonomie nicht genutzt. Basierend auf den Aussagen der SuS und der der Wizards ist das auf die erteilten Aufgaben mit den LEGO Mindstorms-Robotern zurückzuführen. Die genutzte Programmierumgebung und der Schwerpunkt der Aufgaben erzeugte verschiedene Probleme im Bereich der Software. Dadurch schienen die Bereiche Hardware und Umgebung nicht relevant zu sein. Die SuS und die Wizards beschrieben das systematische Vorgehen als hilfreich, um das Problem zu verstehen, beziehungsweise das Vorgehen des Problemlösens. Zu begründen ist dies durch die Abstufung der Problemursachen, die das konkrete Problem immer genauer einschränkt. Von den Wizards wurde eine detaillierte Problemeingrenzung in der Unterkategorie *Programmcode* gewünscht. Diese könnte z. B. eingrenzen, ob die Reihenfolge der verwendeten Programmbausteine fehlerhaft ist oder ein falscher Baustein verwendet wurde. Prinzipiell kann diese Unterscheidung auf der 3. Ebene der Taxonomie vorgenommen werden. Mit der Aufnahme dieses Vorschlags können die 3. und 4. Ebene besser voneinander abgegrenzt werden. Die SuS sowie die Wizards kritisierten, dass sie einige Stufen gerne übersprungen hätten, was ein legitimes Vorgehen war und stringenter kommuniziert werden sollte. Die Verbesserung in der gesamten Einschätzung aller Befragten ist gegebenenfalls auch auf einen Lernprozess im Umgang mit der Taxonomie zurückzuführen. Wenige Wizards beschrieben noch in der 2. Intervention, dass z. B. Probleme mit dem Tool nicht in der Taxonomie enthalten seien. In den weiteren Stunden stuften alle Wizards dieses Problem im Bereich Software ein, nachdem sie sich im Gruppeninterview gegenseitig das Feedback zur geeigneten Kategorisierung gaben. Der Hinweis weiteren Kategorien nachzugehen (z. B. einen falschen Denkansatz) wird im Folgenden nicht in die Taxonomie integriert, da das Ziel der Taxonomie ist, die Problemursachen im Physical-Computing-System aufzuzeigen und nicht allgemein umfassend zu sein. Die Erweiterung der Taxonomie kann gewinnbringend sein, nachdem ihr positiver Einfluss auf das Problemlösen tiefgreifend untersucht ist. Die Ergebnisse dieser Studie sind in Tabelle 5.9 zusammengefasst.

Tabelle 5.9: Zusammenfassung der Evaluation der Hilfestellung *Problemtaxonomie gestuft* durch die WoZ-Studie I

Geräte	LEGO-Roboter, Nao-Roboter
Wizard	sichtbar
Hilfestellung	*Problemtaxonomie gestuft* als direkte Instruktion
Ergebnisse	• SuS beschrieben das Feedback als hilfreich • Betreuer beschrieben das gestufte Feedback als hilfreich und intuitiv • erscheint hilfreich für die Verwendung verschiedener PhC-Geräte (hinsichtlich getesteter Gerätetypen) • die Hilfestellung sollte weitere Kategorien bzw. Abstufungen umfassen • Umgang mit der Hilfestellung benötigt Übung
Schlussfolgerung	Hilfestellung *Problemtaxonomie gestuft* unterstützt den PhC-Prozess, wobei sie zunächst abhängig von den Betreuern ist.

5.4.6 Studiendesign der Wizard-of-Oz-Studie II

An dieser Teilstudie nahmen sechs Schülerinnen und Schüler im Alter von 15 - 17 Jahren teil, die ein berufsorientierendes (9. Klasse) oder wissenschaftliches Praktikum (11. Klasse) absolvierten. Sie bekamen die gleichen Aufgaben wie die SuS der Schülergesellschaft Informatik, jedoch in kondensierter Form. Sie arbeiteten ebenfalls paarweise und füllten die Testinstrumente *Test zu algorithmischen Strukturen* sowie den *Physical-Computing-Test* aus. Diese Studie wurde über vier Tage mit je sechs Stunden durchgeführt, wobei die Testinstrumente als Pre- und Posttest eingesetzt wurden. Dadurch sind die Studien WoZ I und WoZ II vom reinen Zeitaufwand an Lernsituationen und der Intervention miteinander vergleichbar. Der grundlegende Unterschied zur Studie WoZ I besteht darin, dass die SuS das Feedback über den Computer erhielten. Technisch wurde dies wie folgt umgesetzt:

- Die SuS wurden während ihrer Arbeit gefilmt.
- Der Bildschirm der SuS wurde über die Software *Team Viewer* übertragen.
- Die SuS erhielten ein Feedback über Messenger Nachrichten durch *Slack* auf einem bereitgestellten Apple-Tablet.

Die SuS arbeiteten in einem Raum, in dem die Roboter und betreuende Personen anwesend waren. Die BetreuerInnen waren in dieser Stunde nur im Raum, um entweder die Aufgaben an die SuS zu übermitteln, oder technisch zu unterstützen, wenn mit dem Nao-Roboter gearbeitet wurde. Zur Übermittlung der Hilfestellung saß ein Wizard in einem Nebenraum und erhielt die Bildschirmübertragung des Schüler-PCs, das Video der Kamera (über die Interaktion der SuS mit dem Roboter) in Echtzeit und sollte mit einem weiteren PC Nachrichten an die SuS senden (Aufbau in Abbildung 5.4 dargestellt). Diese Nachrichten sollten nach dem gleichen Schema wie in der Wizard-of-Oz-Studie I gegeben werden und entsprechen den zuvor beschriebenen vier Ebenen der Hilfestellung. Um eine Zeitverzögerung des Feedbacks gering zu halten, bekam der Wizard ein vorgefertigtes Dokument an möglichen Unterstützungen, die er nur kopieren musste. Diese Hilfen sind aus den Protokollen der Wizard-of-Oz-Studie I entnommen und entsprechen den Hilfestellungen, die für die gleichen Aufgaben bereits benötigt wurden. Aufgrund des Feedbacks aus der Wizard-of-Oz-Studie I war der Wizard instruiert Stufen des Feedbacks zu überspringen, sofern er es für nötig erachtete. Das galt, wenn der Wizard von den Schülerantworten ableiten konnte, dass diese das Problem bereits verorten können und die Stufe 1 und 2 nicht notwendig sind.
Für die anschließende Auswertung wurden das Video mit der Interaktion der SuS mit dem Roboter und die Log-Daten über die gegebenen Hilfestellungen gespeichert. In regelmäßigen Abständen und an den Stellen, an denen der Wizard intervenierte, wurden Screenshots vom PC der SuS aufgenommen und gesichert, um ihr geschriebenes Programm und mögliche Fehler zu dokumentieren.

5.4.7 Durchführung der Wizard-of-Oz-Studie II

Am ersten Tag der Studie füllten die SuS einen Pretest aus, der aus dem Programmierkompetenztest und dem Physical-Computing-Test bestand. Im Anschluss bekamen sie Aufgaben für die LEGO Mindstorms-Roboter gestellt. Zunächst beschäftigten sie sich mit einfachen Bewegungen der Roboter bis hin zur Kombination von Bewegungen des Roboters abhängig von Sensorwerten des Berührungs- und Gyrosensors. Die Kommunikation der Roboter über Bluetooth wurde im Anschluss behandelt, um z. B. das Verhalten von Tieren als Gruppenverhalten nachzustellen. Dieser Teil diente der Übung und dem Kennenlernen der Roboter und der Programmierumgebung. Am zweiten Tag wurde mit allen Gruppen die Intervention durchgeführt, bei der sie Aufgaben für den LEGO Mindstorms-Roboter gestellt bekamen. Jede Gruppe hatte 1 - 1,5 Stunden Zeit, um die Aufgaben zu lösen (vgl. Abschnitt 3.2.2), und arbeitete einzeln in einem Raum. Am dritten Tag wurden durch die Programmierung eines Chatbots, das Aufnehmen von Keyframes und eine anschließende Kopplung, die Nao-Roboter eingeführt. Dadurch kann der Roboter auf Basis von Befehlen Bewegungen ausführen. Der Interventionsteil zu den Nao-Robotern folge am vierten Tag, an dem die Gruppen erneut einzeln spezielle Aufgaben bearbeiteten, aufgezeichnet wurden und Hinweise bekamen. Für diese Aufgaben hatten die SuS in jedem Fall 1,5 Stunden Zeit und gegebenenfalls zusätzliche 15 min, falls sie kurz vor der Erreichung eines Ziels standen. Der Zeitaufwand wurde höher veranschlagt, da die technischen Gegebenheiten mehr Zeit umfassten, z. B. ein Neuverbinden des Roboters, oder wenn dieser gänzlich neu gestartet werden musste. An einem fünften Tag wurde nur noch der Posttest durchgeführt.

Aufgetretene technische Schwierigkeiten bestanden vor allem darin, dass der verwendete Messenger-Dienst nicht einwandfrei funktionierte. Zwischenzeitlich kam es zu Verzögerungen beim Senden der Nachrichten, so dass die SuS inzwischen selbst auf das zugrundeliegende Problem kamen, oder die Nachricht ankam, jedoch keine Push-Nachricht erzeugt wurde und kein Ton als Notifikation abgespielt wurde. Die SuS wurden zum Beginn der Intervention darauf hingewiesen, dass sie die Nachrichten auf dem Tablet regelmäßig überprüfen sollen. Wenn für die anwesende Lehrkraft anzunehmen war, dass die SuS bereits eine neue Nachricht erhalten haben sollten, erinnerte die Lehrkraft sie daran, die Nachrichten zu überprüfen. Dies geschah mit dem Hinweis, dass die Nachrichten verzögert oder ohne Notifikation ankommen können. Dieses Problem trat circa bei einem Drittel der versandten Nachrichten auf. In einigen Situationen ist nicht eindeutig erkennbar, ob die SuS den Messenger-Dienst auf dem Tablet geöffnet hatten (kein Ton erklingt) und die Nachricht ohne eine erkennbare Reaktion lasen. Es ist ebenfalls möglich, dass die SuS die Nachricht tatsächlich nicht bemerkten. Ein Einfluss auf die Studienergebnisse ist jedoch weitgehend auszuschließen, da die Studie die Reaktion der SuS auf erhaltene Hilfestellungen in Form von Nachrichten und deren inhaltliche Umsetzung untersuchte.

5.4.8 Evaluation der Wizard-of-Oz-Studie II

Da es sich um eine kurze Intervention handelte, wurden die Testhefte von Pre- und Posttest nicht genutzt, um einen Lernerfolg abzuleiten. Sie dienen nur als Orientierung zum Stand der Kenntnisse der SuS. Im Bereich der Programmierkompetenz (vgl. Anhang D.5) beantworteten die SuS 73 % der Testitems richtig. Von den erreichbaren sieben Punkten erreichten sie durchschnittlich fünf Punkte ($MW = 5,2$ und $M = 5$), wobei innerhalb eines Testhefts $Min = 2$ und das $Max = 7$ an erreichten Punkten ermittelt wurde. Das deutet darauf hin, dass die Gruppe im Vergleich zur Wizard-of-Oz-Studie I bessere Programmierkenntnisse aufwies. Generell wiesen die SuS dieser Studie einen höheren Altersdurchschnitt auf, mit dem dies zusammenhängen kann. Im Posttest (vgl. Anhang D.7) veränderte sich die durchschnittlich erreichte Punktzahl nicht, lediglich das Minimum stieg an ($Min = 4$). Die Ergebnisse des Physical-Computing-Tests zeigen im Pre-und Posttest keine auffälligen Veränderungen auf, was ebenfalls zu erwarten war (vgl. Anhang D.11 und D.12). Sie zeigen jedoch etwas bessere Voraussetzungen der SuS im Gegensatz zur Wizard-of-Oz-Studie I auf. Im Bereich der Zuordnung von Input (88 % Fragen richtig beantwortet), Processing (88 %) und Output (79 %) sind sie miteinander vergleichbar. Bei der Bestimmung von Fehlermöglichkeiten und der Zuordnung zu den Problemursachen zeigen die SuS im Bereich Software eine Tendenz, besser abgeschnitten zu haben (mit 67 %) als SuS der WoZ-Studie I (40 %). Es ist anzunehmen, dass diese Ergebnisse mit ihren fortgeschrittenen Programmierfähigkeiten im Zusammenhang stehen. Im Bereich Hardware (67 %) und Umgebung (56 %) zeigen sich keine Auffälligkeiten im Vergleich der beiden Studien. In der Selbsteinschätzung ihrer Vorkenntnisse schätzten sich die SuS der WoZ-Studie II vor und nach der Intervention mit einem $M = 3$ schlechter ein als die SuS der WoZ-Studie I (mit jeweils $M = 4$).

Zur qualitativen Auswertung dieser Studie werden ausschließlich die Reaktionen der SuS und die damit verbundene Diskussion untereinander betrachtet. Nach dem Vorbild von Tsovaltzi et al. (2008) wird untersucht, wie die SuS auf den Wizard reagieren, wobei die positive und negative Reaktion deduktiv übernommen und weitere Reaktionen auf den Wizard während der Analyse in das Codesystem aufgenommen wurden. Dabei wird die Situation in genau eine Kategorie eingeordnet. In dieser Studie wurden die SuS nicht zur Hilfestellung befragt und sie bekamen ausschließlich die Unterstützung vom Wizard, wobei sie nicht wussten, wer diese Nachrichten sendet. Aufgrund von aufgetretenen technischen Schwierigkeiten wurde die Anzahl erhoben, wie oft der Nachrichtenton erklang und die SuS die Möglichkeit hatten, die Nachricht unmittelbar zu bemerken. Um einen Überblick über die vorliegenden Problembereiche zu geben, wurde ebenfalls die Aufgabe, bei der der Wizard Unterstützungen gab, sowie die Ebene in der Hilfestellung betrachtet, in die die Hilfe einzuordnen ist. Für die Datenanalyse wurden insbesondere die Abschnitte des Transkripts untersucht, wenn der Wizard eine Nachricht mit einer Hilfestellung an die SuS sandte. In Tabelle 5.10 ist das ermittelte Kategoriensystem der Schülerreaktionen auf den

Wizard sowie die zugehörige Quantifizierung angegeben.

Tabelle 5.10: Codesystem der Schülerreaktionen in der Wizard-of-Oz-Studie II

Code-Bezeichnung	**Anzahl**
Hilfe für die Aufgabe	
Lücke finden	3
Um eine Kiste fahren	19
Ball greifen	11
Kleineren Ball greifen	2
Malen mit Stift	19
Malen durch Sprachsteuerung	3
Ballerkennung	25
Hilfe auf Stufe	
Stufe 1	3
Stufe 2	23
Stufe 3/4	56
Reaktion auf den Wizard	
Verstehen die Nachrichten nicht und fragen L	2
Nur ein/e PartnerIn will der Hilfe nachgehen	1
Keine Reaktion auf den Wizard	
Bekommen die Nachricht nicht mit	11
Aufgabe im gleichen Moment gelöst	1
Lesen Nachricht nicht vor/besprechen sie nicht	3
Neutrale Reaktion auf den Wizard	
Beschreiben, dass sie die gleiche Idee hatten	1
Eine/r liest Nachricht, teilt sie nicht mit	1
Stellen fest, dass sie das bereits getan haben	3
Lesen Nachricht vor, diskutieren nicht	11
Negative Reaktion auf den Wizard	
Reagieren verstimmt, wenden Hilfe trotzdem an	1
Positive Reaktion auf den Wizard	
Wenden die Hilfe direkt an	32
Wenden die Hilfe später an	5

In der Zählung verschiedener Kategorien fallen geringe Diskrepanzen zwischen der Anzahl von gegebenem Feedback an die SuS ($N = 82$) und die konkrete Beschreibung der Reaktion der SuS auf ($N = 72$). Der Unterschied ist damit zu begründen, dass in einigen Situationen nicht eingeschätzt werden konnte, ob die SuS eine Nachricht bemerkten. Zumeist saßen sie direkt vor dem Tablet und es konnte nicht mit Sicherheit eingeschätzt werden, ob sie auf den Bildschirm des Computers oder auf das Tablet schauten. In Abbildung 5.10 ist zu

erkennen, dass die Verteilung von Problemen mit dem Messenger-Dienst sehr konzentriert auf einzelne Gruppen war. Das Problem konnte zumeist schnell erkannt werden, da die Lehrkraft, die die Aufgaben an die SuS gab, von dem Wizard benachrichtigt wurde, wenn Probleme auftraten. Die Lehrkraft bekam absichtlich nicht die Nachrichten der SuS zugesandt, um das Feedback nicht zu beeinflussen und die Bedingungen eines Tutorensystems realistisch nachzustellen.

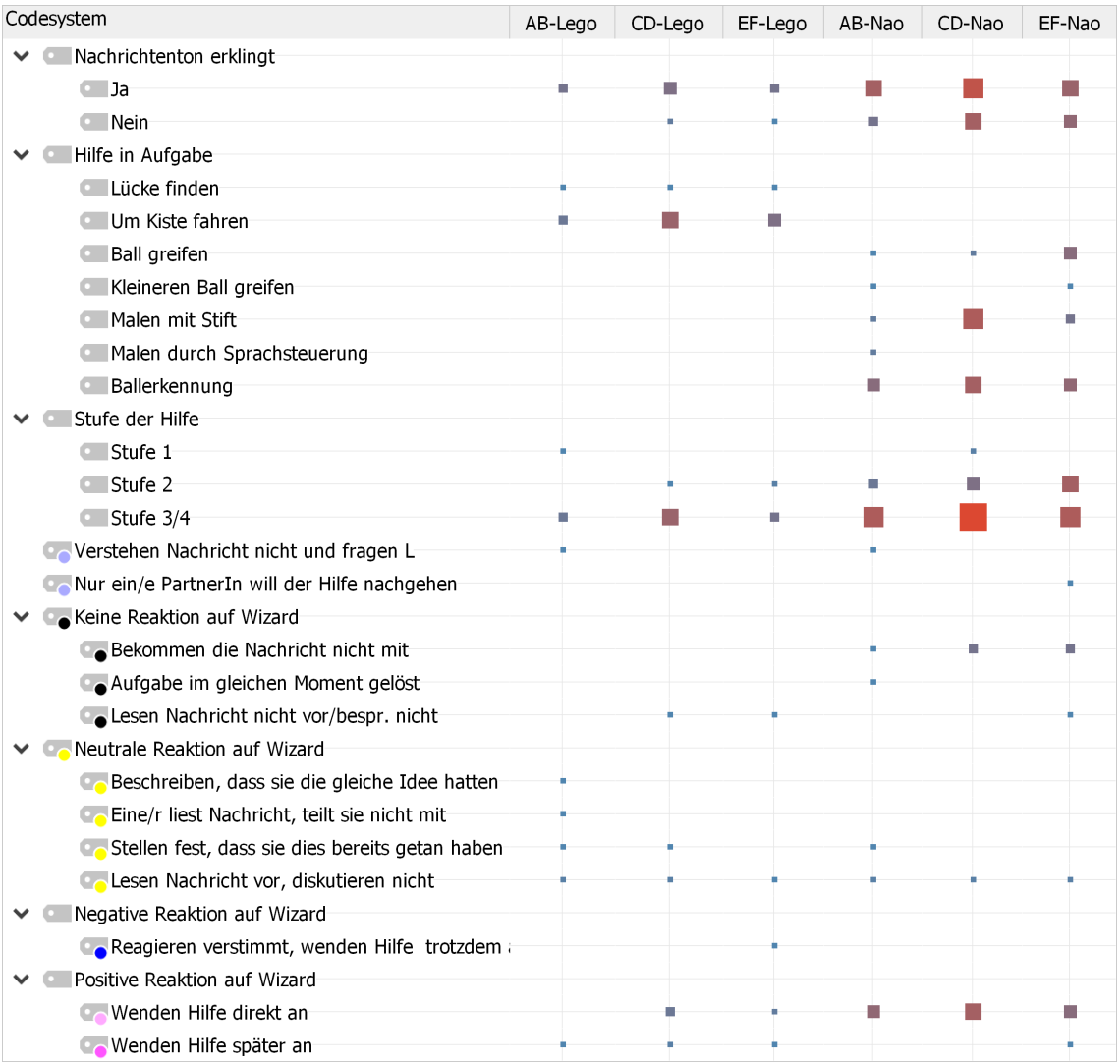


Abbildung 5.10: Code-Matrix der Wizard-of-Oz-Studie II

Bei der Codierung der benötigten Hilfestellung für die verschiedenen Teilaufgaben fällt auf, dass sich die Probleme bei der Interaktion mit den LEGO Mindstorms-Robotern auf das Umfahren einer Kiste konzentrierten. Um die Aufgaben mit Nao-Robotern zu bewältigen, gab der Wizard insbesondere beim Malen mit einem Stift und der Aufgabe zur Ballerkennung Hinweise an die SuS. Eine Aufgabe, in der das zu malende Symbol durch eine Spracheingabe eingegeben werden sollte, wurde nur einer Gruppe erteilt, da diese alle bis dahin erteilten Aufgaben zügig löste. Zwei von drei Gruppen schafften es nicht, die Auf-

gabe zur Ballerkennung zu lösen, da die Interventionsdauer für sie zu kurz war.
Die meisten Hilfestellungen wurden auf Stufe 3 und 4 erteilt (mit $N = 56$), wobei diese nicht genauer unterschieden wurden. Diese gemeinsame Betrachtung hängt damit zusammen, dass diese Ebenen im Feedback inhaltlich ähnlich sind, aber auch für einige Probleme nur drei sinnvolle Stufen gefunden wurden. Hilfestellungen auf der 2. Stufe ($N = 23$) und der 1. Stufe ($N = 3$) wurden den SuS hingegen deutlich seltener gegeben. Das ist insbesondere auf das Feedback der SuS in der Wizard-of-Oz-Studie I zurückzuführen, in der sie anmerkten, dass ihnen die Einordnung eines Problems auf der 1. und 2. Ebene zumeist bewusst war.
Im Folgenden werden die Reaktionen der SuS auf den Wizard dargestellt. Sie werden in folgende Gruppen eingeteilt: keine Reaktion, neutrale Reaktion, negative Reaktion und positive Reaktion. Diese Kategorien sind angelehnt an Tsovaltzi et al., jedoch fokussierten diese sich auf positive und negative Reaktionen auf den Wizard. Da das Ziel der Studie ist, die Effektivität im Umgang mit der Hilfestellung *Problemtaxonomie gestuft* zu ermitteln, wurden ebenfalls *neutrale Reaktionen* codiert. Die Kategorie *keine Reaktion* wurde hinzugenommen, da sie ermitteln lässt, wie präsent die Hilfestellung war. Wenn die SuS nicht auf eine Hilfestellung reagieren, kann dies damit zusammenhängen, dass die Hilfestellung nicht ansprechend genug ist oder bei den SuS in Vergessenheit gerät. Es konnten zwei weitere Formen von Reaktion auf den Wizard beobachtet werden. Dabei handelt es sich einerseits um den Fall *Verstehen die Nachrichten nicht und fragen L.* Dieser Fall trat nur zweimal auf und bezeichnet das Verhalten von SuS, wenn sie Rückfragen an die im Raum anwesende Lehrkraft stellten. Das deutet darauf hin, dass die Nachricht unverständlich formuliert war und die SuS davon ausgingen, dass die Lehrkraft selbst die Hinweise senden würde und der Wizard sei (vgl. Beispiel 5.4.14).

Beispiel 5.4.14 *WoZ II, AB LEGO-Roboter [02:00 Uhr] Nachrichten-Ton erklingt:*
⟶ Fahrzeit überprüfen, um an der Kiste vorbeizufahren.
[B liest die Nachricht und schweigt. A versucht erneut den Roboter umzusetzen und verschiebt den Kistenturm.] [...]

B: *Ach so ja: Fahrzeit überprüfen, um an der Kiste vorbeizufahren.*

A an L: *Das heißt, wir können den jetzt losfahren lassen und anschauen, wie lange er braucht und dann sagen, nach der Zeit soll er links abbiegen?*

L: *Hm, ich möchte jetzt nicht mit euch die Nachricht interpretieren.*

A: *Wir dachten halt, wir sollen mit Hilfe des Sensors merken, wann er links rein muss. War das gar nicht die Aufgabe?*

In dem anderen Fall *Nur ein/e PartnerIn will der Hilfe nachgehen* schlug ein Partner vor, der Hilfestellung zu folgen, wobei der Andere zunächst einer anderen Idee nachgehen wollte, was die Nachricht des Wizards in den Hintergrund geraten ließ.

Die Kategorie *Keine Reaktion* auf den Wizard wird in diesem Abschnitt dargestellt. In diesem Fall war anhand der Schülerreaktion nicht zu erkennen, ob sie mit der Hilfestellung arbeiten konnten und wie sie sie bewerteten. Eine Subkategorie davon ist *Bekommen die Nachricht nicht mit* ($N = 11$), was zumeist eine Folge davon war, dass kein Signalton von dem Messenger-Dienst generiert wurde. Eine weitere Subkategorie ist *Aufgabe im gleichen Moment gelöst*, wenn die Hilfestellung überflüssig wurde, da die SuS die Aufgabe zeitgleich lösten ($N = 1$). Eine weitere Ausprägung, von der keine Reaktion der SuS abgeleitet werden konnte, ist *Lesen Nachricht nicht vor/besprechen sie nicht* ($N = 3$). In diesem Fall wussten die SuS, dass sie eine Nachricht erhalten haben, wobei sie diese weder lasen noch diskutierten. Ein Grund dafür kann sein, dass sie in diesem Moment ihrem eigenen bereits begonnenen Lösungsweg verfolgen wollten.

Als neutrale Reaktionen auf den Wizard wurden die folgenden Kategorien eingeteilt: Die SuS *Beschreiben, dass sie die gleiche Idee hatten* ($N = 1$). Aufgrund dessen ist davon auszugehen, dass der Hinweis im Allgemeinen hilfreich war, jedoch zu diesem Zeitpunkt nicht mehr benötigt wurde. Der Fall *Einer liest Nachricht, teilt sie nicht mit* kam lediglich einmal vor. Für diesen Fall wären zukünftig Regeln für den Wizard oder ein Tutorensystem notwendig, um die SuS bei der Kollaboration zu unterstützen (vgl. Kapitel 5.1).
Ebenfalls neutral zu bewerten ist die Kategorie *Stellen fest, dass sie das bereits getan haben* ($N = 3$). Dabei handelt es sich um sehr kurze Gesprächspassagen, wie in Beispiel 5.4.15 dargestellt.

Beispiel 5.4.15 *WoZ II, Gruppe CD LEGO-Roboter:*
[*C und D lesen die Nachricht still.*]

C: *Ja, das haben wir ja gemacht.*

Die letzte und häufigste Reaktion in diesem Bereich ist *Lesen Nachricht vor, diskutieren nicht* mit $N = 11$. In Beispiel 5.4.16 ist ein Szenario dargestellt, in dem die Reaktion nicht klar ersichtlich ist. Die SuS tauschen sich nicht über die Nachricht aus, weshalb nicht zu erkennen ist, ob sie Änderungen in der Software aufgrund des Feedbacks vornahmen. In einigen Fällen wurde die Reaktion als neutral bewertet, da die SuS keine Bewertung vornahmen und ausschließlich die Nachricht wiederholten (vgl. Beispiel 5.4.17).

Beispiel 5.4.16 *WoZ II, Gruppe CD Nao-Roboter:*
[*10:58 Uhr*] *Nachrichten-Ton erklingt nicht:*
$\longrightarrow$ *Software/Hardware als Bild gesandt*
[*C liest die Nachricht leise. D steht auf und liest mit.*]
[*C und D kontrollieren das Programm.*]

[*D geht zum Roboter und testet die Aufgabe erneut. Der Roboter hält den Ball wieder nicht.*]
[*D stellt fest, dass der Armabstand immer noch zu groß ist.*]

Beispiel 5.4.17 *WoZ II, AB LEGO-Roboter:*
Nachrichten-Ton erklingt:
$\longrightarrow$ *Umgebung als Bild gesandt*
[*B reagiert und öffnet die Nachricht.*]

B: *Uih, Software.*

Innerhalb der gesamten Wizard-of-Oz-Studie II wurde lediglich eine negative Reaktion auf den Wizard gefunden und codiert. In der Subkategorie *Reagieren verstimmt, wenden Hilfe trotzdem an* zeigt die Gruppe eine gereizte Reaktion auf die Nachricht des Wizards. Vermutlich hängt es jedoch nicht mit der Qualität des Hinweises, sondern mit der Frustration über die Schwierigkeit der Aufgabe zusammen, da sie die Hilfe trotzdem direkt anwenden (vgl. Beispiel 5.4.18).

Beispiel 5.4.18 *WoZ II, Gruppe EF LEGO-Roboter:*
[*01:14 Uhr*] *Nachrichten-Ton erklingt:*
Software $\longrightarrow$ *zu drehende Winkel überprüfen*
[*E öffnet die Nachricht unmittelbar und liest die Nachricht genervt vor.*]

E: *Ja!*

[*E und F ändern die Winkel. E ist genervt und stöhnt.*]

Zuletzt wird die Kategorie *Positive Reaktion auf den Wizard* untersucht. Es handelt sich dabei um Situationen, in denen die SuS das Feedback des Wizards aufgriffen und ihre Lösung daraufhin anpassten. Der Fall *Wenden die Hilfe direkt an* bildet mit $N = 32$ hier die häufigste Subkategorie. Beispiel 5.4.19 zeigt dafür einen idealtypischen Ablauf, bei dem die SuS die Nachricht sofort bemerken, dem/der GruppenpartnerIn die Nachricht mitteilen und sie direkt anwenden. In diesem Beispiel erscheint eine Diskussion über die Nachricht nicht notwendig zu sein, da sie ein klares Problem benennt und nach der Adaption der SuS die Aufgabe direkt gelöst wurde.

Beispiel 5.4.19 *WoZ II, Gruppe AB Nao-Roboter:*
[*12:44 Uhr*] *Nachrichten-Ton erklingt:*
Software $\longrightarrow$ *Handabstand überprüfen*
[*B geht sofort zum Tablet und liest die Nachricht laut vor.*]
[*A und B speichern erneut den Handabstand.*]
[*A und B lassen den Roboter wieder nach dem Ball greifen, diesmal hält der Roboter den Ball.*]

Von dieser Beobachtung können die folgenden Hypothesen generalisiert werden: *Computergestütztes Feedback wird von Schülerinnen und Schülern angenommen* und *Computergestütztes Feedback verbessert die Problemlösekompetenzen von Schülerinnen und Schülern.* Es sollte ebenfalls überprüft werden, inwieweit ein vollständig automatisiertes Feedback ähnliche Reaktionen hervorruft.
Das Beispiel 5.4.20 zeigt eine Situation, in der die SuS das Feedback annehmen, wobei sie gemeinsam diskutieren, mit welchen konkreten Werten die Adaption gelingen kann.

Beispiel 5.4.20 *WoZ II, Gruppe CD LEGO-Roboter:*
[*12:12 PM*] *Nachrichten-Ton erklingt:*
Software $\longrightarrow$ *zwischen den Tests, ob die Kiste neben dem Roboter ist, kann dieser ruhig etwas weiter fahren.*
[*C und D öffnen unmittelbar die Nachricht und lesen still. D fängt an zu programmieren.*]

C: *Echt jetzt, dann fährt der ja übel weit weg.*

D: *20 cm meinst du ist zu viel?*

C: *Ja ich würde sagen so 15 oder 10.*

[*C und D diskutieren erneut die Winkel und C dreht manuell den Roboter hin und her. D bemerkt, dass die Winkel nicht übereinstimmen. C setzt den Roboter auf den Boden. Dieser fährt los und meistert die Ecken.*]

In wenigen Situationen bekamen die SuS Hinweise, die sie bei der Fehlersuche innerhalb ihres Programms unterstützten. Da bereits bekannt war, dass der Fehler in dem geschriebenen Programm liegt, wurden die SuS dazu angeleitet ihr Programm in kleineren Teilen und nicht als Gesamtes zu testen. Als mögliches Feedback hätte der Wizard auf eine konkrete Box verweisen können, die womöglich fehlerhaft ist. In dem Beispiel 5.4.21 wenden die SuS den Hinweis direkt an.

Beispiel 5.4.21 *WoZ II, Gruppe AB Nao-Roboter:*
[*1:30 Uhr*] *Nachrichten-Ton erklingt:*
Software $\longrightarrow$ *Programm modular testen*
[*B liest die Nachricht laut vor.*]

A: *Ja stimmt, das könnte man machen.*

[*A und B testen das Programm modular.*]

Der letzte Fall, der betrachtet wurde, ist *Wenden die Hilfe später an* mit einem Vorkommen von $N = 5$. In Beispiel 5.4.22 bekommen die SuS zunächst die Anweisung, dass sich der Roboter um 90 Grad drehen sollte, um regelmäßig zu überprüfen, ob die Kiste noch in der Nähe ist. Sie lesen die Nachricht und arbeiten an ihrem angefangenen Vorgehen weiter. Aufgrund von Problemen mit dem Messenger-Dienst erinnerte die anwesende Lehrkraft

die SuS daran, ihre Nachrichten zu überprüfen. Daraufhin sehen sich die SuS die Nachricht erneut an und arbeiten weiter mit diesem Ansatz.

Beispiel 5.4.22 *WoZ II, Gruppe CD LEGO-Roboter:*
[*12:02 Uhr*] *Nachricht-Ton erklingt:*
Software $\longrightarrow$ *um zu sehen, ob genau neben dem Roboter die Kiste ist, muss er sich um 90 Grad drehen.*
[*C und D nicken nur und lassen den Roboter erneut das Programm ausführen.*]

C: *Da wendet er, weil er nichts mehr gesehen hat.*

[*D geht zurück zum PC*]

D: *Vielleicht müssen wir ihn noch ein Stück näher an die Kiste heranfahren lassen.*

L: *Aber lest euch auch die Hinweise durch.*

D: *Ok.* [*D öffnet den Chat und liest still die Nachricht. C sitzt weiterhin bei dem fahrenden Roboter.*]

[*12:04 PM*] *Nachrichten-Ton erklingt nicht:*
Software/Hardware $\longrightarrow$ *90 Grad im Programm sind nicht unbedingt 90 Grad beim Roboter*

C: *Vielleicht muss er sich einfach mal drehen.*

D: *Das können wir machen indem wir den Winkel ändern.*

Aufgrund der Verfügbarkeit des Feedbacks im Chatverlauf ist es den SuS möglich auf Hinweise zu einem späteren Zeitpunkt zurückzugreifen. Auch für andere Gruppen könnte die folgende Hypothese zutreffen: *Durch die Verfügbarkeit von Feedback prüfen die Schülerinnen und Schüler Lösungsansätze mehrfach.*

5.4.9 Diskussion und Zusammenfassung der Wizard-of-Oz-Studie II

Insgesamt ist die durchgeführte Studie als erfolgreich zu bewerten (Zusammenfassung in Tabelle 5.11). Die SuS reagierten zumeist positiv auf den Wizard, indem sie die Hilfestellungen für ihre Problemlösung anwendeten. Wie in Abbildung 5.10 dargestellt, waren die Schülerreaktionen auf den Wizard deutlich positiv und die SuS nutzen sie gezielt, um ihre Aufgaben zu lösen. Das Feedback war zumeist gut verständlich, da die SuS nur in seltenen Fällen nachfragten oder oft dem Feedback entsprechend handelten. Für weitere Studien wäre interessant, inwieweit die SuS die Aufgaben effektiver lösen (vgl. Tsovaltzi et al., 2008) und ihre Problemlösefähigkeiten durch ein Feedback verbessern.
Als Hilfestellungen wurden insbesondere die 3. und 4. Ebene der Hilfestellung verwendet. Im Folgenden sollten diese Abstufungen verfeinert und weitere Stufen erarbeitet werden, damit ein mehrstufiges Feedback erteilt werden kann. Es ist üblich, mit kognitiven Tutorensystemen leistungsheterogene Gruppen zu adressieren. Das scheint jedoch nur möglich

zu sein, wenn das zugrundeliegende kognitive Modell ausdifferenzierter ist.
Ein weiterer Aspekt, der noch nicht berücksichtigt wurde, ist die Unterstützung von Kollaboration der SuS. In einigen Subkategorien wurde bereits deutlich, dass die SuS ungenügend als Team fungierten, da sie sich gegenseitig die erhaltenen Hinweise nicht mitteilten oder sie nicht mit dem Teammitglied diskutierten.
Die vorgestellte Untersuchung bildet eine solide Grundlage, um digitale Technologien wie Tutorensysteme für das Problemlösen im Physical-Computing-Prozess zu entwickeln. Dabei können zukünftig ebenfalls Hilfestellungen genutzt werden, die die SuS beim Durchlaufen des Problemlöseprozesses unterstützen. Dafür kann die Gliederung in Teilaufgaben gewinnbringend sein, da bereits für den Inquiry-Prozess gezeigt wurde, dass kognitive Prozesse parallel zu den erteilten Teilaufgaben ablaufen können (vgl. Patzwaldt et al., 2014).

Tabelle 5.11: Zusammenfassung der Evaluation der Hilfestellung *Problemtaxonomie gestuft* durch die WoZ-Studie II

Geräte	LEGO-Roboter, Nao-Roboter
Wizard	verdeckt
Hilfestellung	*Problemtaxonomie gestuft* als direkte Instruktion
Ergebnisse	• die SuS reagierten zumeist positiv auf das Feedback des Wizards • die SuS nutzten die Hinweise, um die Aufgaben zu lösen • eine detaillierte Abstufung der Ebenen sollte vorgenommen werden
Schlussfolgerung	Hilfestellung *Problemtaxonomie gestuft* ist als Grundlage eines kognitiven Tutorensystems geeignet.

5.5 Zusammenfassung

Dieses Kapitel verfolgt das Ziel, verschiedene Hilfestellungen für SuS in Physical-Computing-Aktivitäten zu evaluieren. In der ersten Untersuchung wurde die Hilfestellung *Evaluationsphase digital* verwendet, die ein Grundgerüst für die Evaluationsphase bietet. Durch die quantitative Analyse wurde sichtbar, dass die Durchführungsphase und die Auswertungsphase der 3. Studie (LEGO-Roboter, Hilfestellung *Evaluationsphase digital*) stärker miteinander korrelierten als in den anderen Gruppen der Studien 1 und 2 (LEGO-Roboter, Hilfestellung *4-Phasen analog*). Das deutet darauf hin, dass die Auswertung von beobachteten Ereignissen öfter durchlaufen wurde. Die qualitative Analyse zeigt Tendenzen, dass die SuS mehr mögliche Problemursachen für ihr Problem in Betracht zogen,

jedoch die meisten Probleme durch die Veränderung von Software behoben. Für den Umgang mit weiteren Physical-Computing-Geräten wie z. B. E-Textilien (vgl. Kafai et al., 2014a) ist es jedoch notwendig, auch innerhalb der anderen Systemkomponenten Fehler suchen zu können. Es erscheint naheliegend, dass die SuS somit ein besseres Verständnis über die Wechselwirkungen innerhalb des Systems entwickeln können. Das Lösen eines Problems kann in den verschiedenen Bereichen der Problemursachen verschieden komplex sein. Die SuS tendierten jedoch dazu, ihre Probleme fast ausschließlich im Bereich der Software zu lösen. Daher kann ein effizientes Problemlösen durch die gezielte Auswahl einer Systemkomponente begünstigt werden.
In zwei weiteren Studien wurde die Hilfestellung *Problemtaxonomie gestuft* evaluiert. Diese wurde als Grundlage für ein kognitives Tutorensystem entwickelt und bietet Lehrkräften eine Strukturierung, um den SuS Feedback über vorliegende Probleme bereitzustellen. Als Vorbild diente die in Kapitel 4 entwickelte Problemtaxonomie, in die Probleme im Bezug auf das Physical-Computing-System eingeordnet werden können. Ähnlich wie in anderen Tutorensystemen beschrieben, erhielten sie dafür ein abgestuftes, detaillierter werdendes Feedback (vgl. Koedinger und Aleven, 2007). Es wurde die Intensität des Feedbacks durch die Eingrenzung des Problembereichs, Textfeedback und teilweise durch konkrete Lösungsvorschläge (vgl. McLaren et al., 2014) differenziert. Die Wizard-of-Oz-Studie I zeigt ein positives Feedback der SuS auf die Hilfestellung. Es traten verschiedene Probleme auf und die meisten SuS beschrieben die Hilfestellung *Problemtaxonomie gestuft* als Unterstützung bei der Bewältigung der Aufgaben. Ebenfalls zeigen die verwendeten Testinstrumente die Tendenz auf, dass die SuS ihre Problemlöse- und Programmierfähigkeiten verbessert haben. Insbesondere Umgebungseinflüsse, die zuvor kaum Berücksichtigung fanden, wurden von den SuS vielfältiger beschrieben.
Die Wizards unterbreiteten diverse Vorschläge, um die Hilfestellung zu verfeinern, was in folgenden Untersuchungen Berücksichtigung finden sollte. Sie bewerteten die Hilfestellung positiv, wobei des Öfteren angemerkt wurde, dass die ersten beiden Stufen nicht benötigt wurden. Insbesondere diese Ebenen sollten im Folgenden untersucht und differenziert werden. Möglicherweise sind die ersten beiden Ebenen sinnvoll, um den SuS die Komponenten eines Physical-Computing-Systems aufzuzeigen, können jedoch bei fortschreitender Erfahrung im Physical Computing übersprungen werden.
Somit scheint die entwickelte Problemtaxonomie und deren Abstufung als Hilfestellung ein geeignetes Mittel zu sein, um SuS ein Feedback über vorliegende Probleme zu geben. Da die Ergebnisse dieser Studie abhängig von den einzelnen Wizards sein können und nicht eindeutig nachzuvollziehen ist, ob die Wizards bewusst oder unbewusst von der Hilfestellung *Problemtaxonomie gestuft* abwichen, wurde die Wizard-of-Oz-Studie II durchgeführt. In der Wizard-of-Oz-Studie II war der Wizard nicht sichtbar und es wurde untersucht, wie die SuS auf das Feedback reagierten und ob es ihnen half, die Aufgaben zu bewältigen. Die SuS zeigten fast ausschließlich positive Reaktionen auf den Wizard. Sie bewerteten die Nachrichten als hilfreich bzw. wendeten sie direkt an. Für zukünftige Tutorensysteme im

Physical Computing sollten die 3. und 4. Ebene der Hilfestellungen weitere Abstufungen umfassen, damit ein mehrstufiges Feedback bei Fortgeschrittenen möglich ist, die bereits die 1. und 2. Ebene überspringen. Eine Implementation des Tutors, der mit dem Roboter kommuniziert, wäre dafür eine Möglichkeit, um ein gezieltes Feedback für die SuS zu erzeugen, ohne dass ihre Aktionen durch einen Menschen beobachtet werden müssen (vgl. Spikol et al., 2016). Es wäre möglich, den Zeitpunkt der Hilfestellungen zu bestimmen, z. B. durch eine Inaktivität der SuS am Physical-Computing-Gerät und der Programmierumgebung. Darüber hinaus könnte die Programmierumgebung ein Feedback erzeugen, wenn die SuS unsystematisch Veränderungen am Programm vornehmen. Für die Evaluation von Hardware- und Umgebungsfaktoren können Sensoren und Aktuatoren des Physical-Computing-Geräts im Hintergrund ausgelesen und interpretiert werden, um im Anschluss ein Feedback an die SuS zu senden.
Die aufgezeigten Untersuchungen lassen zu diesem Zeitpunkt noch keine allgemeinen Rückschlüsse auf den Lernerfolg der SuS zu. Dafür sind validierte Testinstrumente notwendig, die sich auf den Problemlöseprozess im Physical Computing konzentrieren und diverse Altersstufen abdecken, um einen Deckeneffekt zu vermeiden. Die verwendeten Testinstrumente zeigen nur marginale Veränderungen der Schülerkompetenzen auf. Die SuS nutzten jedoch direkte Instruktionen intensiver als prozessuale Unterstützungen. Dabei war ein direkter Zusammenhang des Feedbacks und des Problemlösens durch die SuS erkennbar.
Von Koedinger und Anderson (2013, S. 16) wurden Guidelines zum Design von kognitiven Tutoren beschrieben. Sie formulieren die Schritte „1.) Identify the execution space. 2.) Look for implicit planning in verbal reports. 3.) Model this implicit planning. 4.) Use the model to drive tutor design. 5.) Test the tutor implementation.“ Die ersten der genannten Schritte sind ebenfalls in diesem Kapitel wiederzuerkennen. Die ersten drei Schritte wurden durch Literaturreviews und die Analyse von Problemen durchgeführt und anschließend durch die Beschreibungen der SuS und der Wizards angepasst. Der 4. Schritt wurden in den Wizard-of-Oz-Studien auf unterschiedlichen methodischen und technischen Niveaus vorgenommen. Basierend darauf kann zukünftig eine Implementation des kognitiven Tutors erfolgen. Weitere Iterationen dieses Prozesses sind notwendig, um für die Schule ein nutzbares Unterstützungs-Tool bereitstellen zu können. Die Ergebnisse der Studien sind jedoch bezüglich der Anwendbarkeit des Feedbacks für die SuS vielversprechend.

Kapitel 6

Zusammenfassung und Ausblick

Dieses Kapitel fasst zentrale Ergebnisse dieser Dissertation zusammen und zeigt weitere Forschungslücken und -ansätze für zukünftige Arbeiten auf.

6.1 Zusammenfassung

Physical-Computing-Geräte werden vermehrt in Lernkontexten verwendet und sollen gezielt im Informatikunterricht genutzt werden (vgl. Bildungsserver Berlin-Brandenburg, 2015). Darüber hinaus existieren vielversprechende Ergebnisse bezüglich der Motivationsförderlichkeit von Physical Computing und der Vermittlung informatischer Konzepte durch Physical Computing. Bislang sind die meisten Forschungsergebnisse auf einzelne Geräte fokussiert und der zugrundeliegende Prozess für die Bewältigung von Physical-Computing-Aufgaben steht im Hintergrund und ist ungenügend empirisch untersucht. Konkrete Interventionen zur Unterstützung des Prozesses sind bislang ebenso rar.

Diese Arbeit beschäftigte sich mit den folgenden Forschungsfragen:

1. *Welche Gemeinsamkeiten und Unterschiede weisen die Prozesse Physical Computing und die wissenschaftliche Erkenntnisgewinnung auf? Kann Physical Computing als Arbeitsweise der Erkenntnisgewinnung in der Informatik genutzt werden?*

2. *Welche Probleme treten bei der Interaktion mit Physical-Computing-Geräten auf und wie können diese kategorisiert werden?*

3. *Welche Arten von Hilfestellungen sind effektiv für Schülerinnen und Schüler, um den Physical-Computing-Prozess zu unterstützen?*

Zur Beantwortung der 1. Forschungsfrage wurde ein Literaturreview vorgenommen, das bereits Zusammenhänge von der Arbeitsweise des wissenschaftlichen Experimentierens und dem Physical-Computing-Prozess aufzeigt. Um einen Vergleich dieser Prozesse zu ermöglichen, wurde der Physical-Computing-Prozess aus verschiedenen Beiträgen abgeleitet und in einem Modell zusammengefasst. Ein literaturbasierter Vergleich wurde zuerst

vorgenommen, der insbesondere bei der Betrachtung der Hauptphasen eine Parallelität beider Prozesse aufzeigt. Dieses Ergebnis ermöglicht die Codierung des Physical-Computing-Prozesses in den literaturgeleiteten Hauptphasen, die zugleich den Hauptphasen des wissenschaftlichen Erkenntnisgewinnungsprozesses entsprechen. Eine Ausnahme besteht darin, dass je nach Modell des (SI-)Prozesses die Phase der Experimentdurchführung, im Physical-Computing-Prozess der Implementations- und Durchführungsphase entspricht.
Dieses Modell wurde durch verschiedene qualitative Studien empirisch weitgehend gestützt und in Teilen adaptiert. Basierend auf dem angepassten Physical-Computing-Prozess wurden daraufhin Gemeinsamkeiten und Unterschiede der Gesamtprozesse, der Aktivitäten innerhalb der Prozessphasen und der Phasenübergänge abgeleitet. Die Prozesse weisen insgesamt viele Gemeinsamkeiten bei der inhaltlichen Betrachtung der Phasen und den zugehörigen Teilphasen auf. Der Verlauf des Gesamtprozesses verläuft iterativ und diverse Phasen werden übersprungen. Konkrete Unterschiede sind in der Implementationsphase des Physical-Computing-Prozesses zu finden, in der Software neben der Hardwarekonstruktion implementiert wird. Die Motivation zum Beginn des Prozesses kann unterschiedlich sein und geht im Physical Computing seltener von einem beobachteten Phänomen aus. Diese Arbeit beschreibt ein Physical-Computing-Prozessmodell, das als Grundlage für weitere Forschung im Physical Computing genutzt werden kann. Darüber hinaus eröffnet es weitere Möglichkeiten, die Übertragbarkeit von Forschungsergebnisse der wissenschaftlichen Erkenntnis auf die Informatik zu untersuchen.

Die 2. Forschungsfrage untersuchte Probleme und zugehörige Problemursachen, die bei Aktivitäten mit Physical-Computing-Systemen auftreten können. Das Wissen über konkrete Probleme wird benötigt, um anschließend Hilfestellungen abzuleiten und didaktisch reduzieren zu können. Basierend darauf kann z. B. untersucht werden, inwieweit der Einfluss von einzelnen Bereichen der Problemursachen reduziert werden kann, um die Komplexität des Gesamtprozesses zu minimieren.
Zur Untersuchung dieser Fragestellung wurden die Videodaten und Transkripte recodiert, die im Rahmen der 1. Forschungsfrage erhoben wurden. Die Untersuchungskategorien wurden literaturbasiert und auf Grundlage der definierten Systemkomponenten eines Physical-Computing-Systems gebildet. Ein zentrales Ergebnis ist, dass die Systemkomponenten Software, Hardware und Umgebung die Hauptproblemursachen sind, jedoch Überscheidungen dieser Bereiche häufig auftreten. Als eine weitere Problemursache wurden verschiedene Aspekte von mathematischen und physikalischen Grundlagen ermittelt. Zusammenfassend wurde eine Problemtaxonomie erstellt, die als Grundlage zur Beschreibung von Problemursachen für den Bereich Physical Computing dient und zukünftig erweitert werden kann.

Die 3. Forschungsfrage verfolgte das Ziel verschiedene Hilfestellungen zu evaluieren, die den Physical-Computing-Prozess unterstützen könnten. Dafür wurde zunächst eine Hil-

festellung untersucht, die den Physical-Computing-Prozess für die SuS vorstrukturieren sollte. Es wurde beobachtet, dass die SuS diese Hilfestellung kaum nutzten und sich nur marginale Auswirkungen auf den Prozess zeigten. Basierend auf der entwickelten Problemtaxonomie und existierenden Forschungsergebnissen der kognitiven Tutorensysteme wurde ein regelbasiertes und mehrstufiges Feedback-Modell entwickelt und erprobt. Die SuS, die Instruktionen auf Grundlage des Modells bekamen, berichteten, dass das erteilte Feedback hilfreich war, um ihre Probleme zu verstehen und lösen zu können. Damit bildet das entwickelte Feedback-Modell eine Grundlage für weitere Untersuchungen zur Feedbackgenerierung in Physical-Computing-Aktivitäten.

Konkrete Einflüsse auf die Schulpraxis

Von den Ergebnissen dieser Dissertation können konkrete Auswirkungen auf die Schulpraxis abgeleitet werden. Somit verbleiben die Resultate nicht als reiner Forschungsgegenstand, sondern können gezielt im Informatikunterricht verwendet werden.
Abgeleitet von der 1. Forschungsfrage können Maßnahmen ergriffen werden, die den Prozess des Physical Computing unterstützen. Aus der Forschung zur wissenschaftlichen Erkenntnisgewinnung ist bekannt, dass in dem Problemlöseprozess die kognitiven Prozesse parallel zu gestellten Teilaufgaben verlaufen (vgl. Patzwaldt et al., 2014). Daher können Teilaufgaben im Physical Computing genutzt werden, um die SuS durch den Prozess zu leiten.
Im Rahmen der 2. Forschungsfrage wurde ein Modell entwickelt, das Problemursachen im Physical Computing beschreibt. Dieses Modell kann den SuS zur Verfügung gestellt werden, um die Problemursachen sichtbar und bewusst zu machen. Des Weiteren zeigt es Überschneidungen der Bereiche von Problemursachen auf, was den Handlungsspielraum zum Lösen eines Problems illustriert. Diese Maßnahmen können die kognitive Beanspruchung für SuS sowie Lehrkräfte reduzieren.
Die 3. Forschungsfrage untersuchte den Einsatz verschiedener Hilfestellungen zur Unterstützung des Physical-Computing-Prozesses. Die SuS und betreuenden Personen beschrieben die gestufte Hilfestellung basierend auf der Problemtaxonomie als hilfreich und intuitiv. Die aufgezeigten Regeln können Lehrkräfte nutzen, um den SuS ein Feedback über vorliegende Probleme im Unterricht zu geben. Auch die Verwendung von Karten mit Hilfestellungen, wie sie bereits in den Naturwissenschaften vorgeschlagen wurden (vgl. Arnold et al., 2014), können bei der Bewältigung von Problemen im Physical Computing aufgrund der Ähnlichkeit der Prozesse unterstützend auf den Physical-Computing-Prozess wirken. In der Wizard-of-Oz-Studie II wurden Probleme bei der Kollaboration der SuS ermittelt. Beispielsweise besprachen die SuS eigene Beobachtungen oder das Feedback des Wizards nicht miteinander. In den Schulen sollten daher vor oder während des Physical-Computing-Prozesses Hinweise gegeben werden, wie Probleme kollaborativ gelöst werden können (vgl. Walpuski und Sumfleth, 2007).

Limitationen

Die vorgestellten Ergebnisse unterliegen einigen Limitationen, die zum Teil bereits in den jeweiligen Kapiteln thematisiert wurden. Im Folgenden werden die limitierenden Faktoren dieser Arbeit zusammengefasst und kurz diskutiert.
Alle Schülerinnen und Schüler, die an den Studien teilnahmen, interessierten sich bereits für die Informatik. Das kann sich positiv auf die Motivation und Volition der SuS auswirken, was in diesem Zusammenhang bedeutet, dass sie bessere Leistungen erzielen können und zunehmend gewillt sind die gestellten Aufgaben zu lösen. Das kann den Problemlöseprozess sowie den Umgang mit den gegebenen Hilfestellungen positiv beeinflusst haben. Ein Einfluss auf die Ergebnisse ist jedoch als gering einzuschätzen, da SuS zumeist motiviert sind, wenn sie mit Robotern im Unterricht arbeiten können (vgl. Cross et al., 2015).
Als Untersuchungsgegenstand wurden den SuS ausgewählte Physical-Computing-Geräte bereitgestellt und darauf abgestimmte Aufgabenstellungen erteilt. Die Verwendung von drei verschiedenen Geräten deckt jedoch nicht die Bandbreite von Physical-Computing-Geräten ab, wobei darauf geachtet wurde, Roboter und Mikrocontroller für die Untersuchungen zu nutzen. Dadurch wurden zwei häufig genutzte Gerätetypen untersucht, um ein möglichst breites Spektrum abzudecken. Nichtsdestotrotz sind weitere Geräte, wie z. B. Mikrocontroller mit bereits angelöteten Sensoren und Aktuatoren wie der *BBCmicro:bit*[1] und *Calliope Mini*[2] verbreitet und gelangen zunehmend in die Schulen. Abhängig von den Gerätespezifikationen können bisher nicht betrachtete Probleme auftauchen bzw. kann die Häufigkeit der aufgetretenen Probleme in den Bereichen von Problemursachen unterschiedlich verteilt sein (vgl. Kapitel 4). Des Weiteren sind Einflüsse auf den Ablauf des Physical-Computing-Prozesses möglich, wie es bereits bei der Arbeit mit Arduino-Mikrocontrollern ersichtlich wurde. Da das Programm direkt (ohne einen Ortswechsel) hochgeladen und ausgeführt wird, kann diese zeitliche Verzögerung bei dem Erhalt und der Umsetzung des Feedbacks Probleme bereiten. Da die Auswahl der Problemursachen jedoch von der Definition des Physical-Computing-Begriffs ausgeht, ist davon auszugehen, dass die Problemtaxonomie über verschiedene Aufgaben und Gerätetypen hinweg stabil ist. Die Ergebnisse der 2. Forschungsfrage (vgl. Kapitel 4) deuten ebenfalls darauf hin. Ein weiterer Aspekt, der bislang nicht betrachtet wurde, ist die Auswirkung der Programmierumgebung auf den Problemlöseprozess und konkrete Probleme. Bislang arbeiteten die SuS mit blockbasierten Programmiersprachen, wobei gerade in der Programmierung erfahrene SuS den Wunsch äußerten mit der Arduino IDE[3] codebasiert zu arbeiten, da sie sich mit dieser bereits auskannten. Diesbezüglich ist eine Reduzierung der Anzahl von Softwareproblemen zu erwarten.
Bevor die Interventionen vorgenommen wurden, absolvierten alle SuS eine Einführung in

[1] http://microbit.org/de/
[2] https://calliope.cc
[3] https://www.arduino.cc/en/Main/Software

die Programmierumgebung und die Physical-Computing-Geräte. Sie bearbeiteten einige Aufgaben, die sie auf die Intervention vorbereiteten, ohne jedoch konkrete Probleme vorwegzunehmen. Bei dem Umgang mit der Programmierumgebung und der Firmware des Roboters ist die Reduzierung von Problemen naheliegend, wenn die SuS über einen langen Zeitraum damit interagieren.
Aufgabenstellungen sowie Hilfestellungen wurden von einer Betreuungsperson bzw. verschiedenen Wizards in den Studien erteilt. Ein Einfluss der Betreuungspersonen ist daher nicht auszuschließen. Insbesondere der Studienteilnehmer, der ohne Partner arbeitete, wurde von der Betreuungsperson möglicherweise stärker beeinflusst. Der Proband wurde mehrfach aufgefordert seine Gedanken zu äußern, da er nicht gemeinsam mit einem Gruppenmitglied diskutieren konnte. Bei der Durchführung der Wizard-of-Oz-Studien ist der Einfluss der verschiedenen Wizards zu berücksichtigen. Sie handelten regelbasiert, jedoch ist nicht im Detail nachvollziehbar, ob bewusst oder unbewusst weitere Informationen an die SuS von einem Wizard gegeben wurden, was z. B. andere Wizards nicht getan haben. Der Einsatz von mehreren Wizards ist jedoch hilfreich, um das System möglichst unabhängig von dem Wizard zu testen (vgl. Höysniemi et al., 2004). In der WoZ-Studie I wurde das Feedback von bis zu fünf verschiedenen Wizards erteilt. Darin wurden in der 2. und 3. Intervention dieser Studie fast ausschließlich positive Reaktionen auf die Problemtaxonomie als Hilfestellung festgestellt. Deshalb ist anzunehmen, dass der Einfluss der Wizards auf die Studienergebnisse gering ist.
Ein Einfluss der Nutzung von Videografie auf die SuS und eine damit begründete Verhaltensänderung ist weitgehend auszuschließen. Die Kameras wurden im Raum positioniert, so dass sie zumeist über die Schultern der SuS filmten oder entfernt am Rand standen. Dadurch sollten sie im Bewusstsein der SuS in den Hintergrund geraten. Es wurde darauf verzichtet eine Lampe an der Kamera leuchten oder blinken zu lassen, die die Aufnahme verdeutlicht. Des Weiteren wurde von der Versuchsleitung die Kamera lediglich zum Start der Intervention angeschaltet und anschließend die Bildeinstellungen nicht mehr überprüft, um es für die SuS aus dem Fokus zu rücken. Da die SuS selten zur Kamera schauten oder diese erwähnten, ist davon auszugehen, dass sie davon nur geringfügig beeinflusst wurden.
Das kollaborative Setting bei der Bearbeitung der Aufgaben verbesserte möglicherweise die Problemlösefähigkeiten innerhalb der Gruppen. Walpuski und Sumfleth (2007) beschreiben die Verknüpfung von Inquiry-Aufgaben und Kollaboration als besonders vielversprechend. Ein wichtiger Aspekt dabei ist, dass die SuS sich in ihrem Wissen und ihren Fähigkeiten ergänzen können und somit zu einem Ziel gelangen, obwohl einzelne Schülerinnen oder Schüler nicht wissen, wie sie weiterarbeiten sollen. Darüber hinaus kann die Kollaboration motivationsfördernd wirken, wozu diverse Belege existieren (vgl. Azmitia und Montgomery, 1993; Maldonado et al., 2009). Aufgrund von technischen Schwierigkeiten in der Durchführung der Wizard-of-Oz-Studie II ist der Einfluss auf die Studienergebnisse zu diskutieren. In einem Drittel der Fälle ertönte das durch einen Wizard versandte Feedback nicht. Es ist ebenfalls nicht zweifelsfrei feststellbar, ob die SuS die Nachricht mitbekamen

und sie nicht vorlasen oder ob sie sie nicht bemerkten. Bei der Evaluation der Studienergebnisse wurden nur die Fälle betrachtet, in denen zu erkennen war, dass die SuS die Nachricht bemerkten. Hätte das Ertönen des Nachrichtentons zu 100 % funktioniert, so ist davon auszugehen, dass die SuS weniger frustriert gewesen wären, sobald sie bei der Bearbeitung der Aufgabe nicht vorankommen. Aufgrund der sehr positiven Reaktionen auf die Hilfestellung ist jedoch nicht von einem Einfluss auf die Studienergebnisse auszugehen.

6.2 Ausblick

Die Ergebnisse dieser Dissertation bilden eine wichtige Grundlage für weitere Forschung im Bereich des Physical Computing und als MINT-didaktischer Ansatz für gemeinsame Problemlöseprozesse. Darüber hinaus werden in diesem Abschnitt konkrete Ansätze für die Weiterentwicklung von Physical-Computing-Projekten im schulischen Unterricht vorgestellt.

Systematische Analyse von Physical-Computing-Geräten

Aufgrund der bislang auf Geräte konzentrierten Forschung sollte eine Kategorisierung von Physical-Computing-Geräten auf ihre Merkmale hin vorgenommen werden. Bei der Bewältigung von Design-Aufgaben wurde bereits festgestellt, dass Lernen kontextabhängig und gerätespezifisch sein kann (Crismond, 2001). Mit einer Kategorisierung von Physical-Computing-Geräten können Ergebnisse innerhalb der Geräte-Kategorien übertragen werden und zukünftige Geräte benötigen keine weiteren Analysen auf bereits untersuchte Aspekte innerhalb der Kategorie. Dazu gehört ebenfalls ein kritischer Umgang mit den Medien (Physical-Computing-Geräten) hinsichtlich ihrer Eignung für zu fördernde Kompetenzen in spezifischen Altersstufen. Erprobungen der inhaltlichen Anwendungsbereiche und möglichen didaktischen Reduktionen sollten vorgenommen werden. Dadurch wäre es möglich ein Physical-Computing-Curriculum zu entwickeln, welches in seiner Komplexität abgestuft ist und trotzdem unabhängig von speziellen Geräten bleibt.

MINT-didaktischer Ansatz

Ein Gegenstand dieser Arbeit war ein fächerverbindender Ansatz, um Physical Computing mit erforschten Prozessen der Naturwissenschaften zu verbinden. Literaturgeleitet und empirisch konnten vielversprechende Gemeinsamkeiten der Naturwissenschaften und der Informatik gezeigt werden. Die Entwicklung und Untersuchung eines MINT-Inquiry-Prozesses erscheint für interdisziplinäre Unterrichtskontexte geeignet zu sein, um den Fokus der Kompetenzentwicklung auf mehrere Fächer zu verlagern und fächerverbindendes Lernen zu ermöglichen. Dafür können zunächst gezielte Phänomene bzw. Lerngegenstände mit interdisziplinärem Charakter gewählt und von ihnen eine konkrete Prozessbeschreibung induktiv abgeleitet werden. In einem Design-Based-Research-Ansatz ist anschließend

eine Anpassung des Prozesses an Probleme und reale Bedingungen in schulischen Lernszenarien möglich.

Validierte Messinstrumente für Physical-Computing-Aktivitäten

Wie in anderen MINT-Fächern üblich ist auch in der Informatikdidaktik der Einsatz von validierten Testinstrumenten wünschenswert. Ein Vorschlag für die Erhebung von Problemlösekompetenzen in Physical-Computing-Aktivitäten ist bereits in dieser Arbeit enthalten. Als eine besondere Herausforderung erweisen sich dabei die diversen Vorkenntnisse der SuS. Daher zeigte sich bei der Pilotierung des Testinstruments ein Deckeneffekt. Für eine fein abgestufte Erhebung der Problemlösekompetenzen und die Möglichkeit Verbesserungen durch ein Pre-Posttestdesign zu ermitteln, sollten die folgenden Maßnahmen ergriffen werden:

- Erweiterung des Testinstruments durch weitere Aufgabenbereiche und Items,
- Testitems sollten ein größeres Spektrum an Physical-Computing-Geräten abdecken,
- Einschränkung der adressierten Klassenstufe, abhängig von benötigten Vorkenntnissen.

Nach den Adaptionen des Testinstruments sollte es zunächst in Schülergruppen pilotiert werden, die mehrwöchige Physical-Computing-Aktivitäten, z. B. im Rahmen von Schülergesellschaften und Arbeitsgemeinschaften, durchführen. Dadurch soll der Deckeneffekt beim Einsatz in Regelklassen minimiert werden.

Automatisiertes Feedback

Lehrkräfte stehen nicht ständig zur Verfügung und SuS müssen warten, bis die Lehrkraft Zeit hat ihnen weiterzuhelfen. Darüber hinaus sind die Lehrkräfte gegebenenfalls keine ExpertInnen auf dem Gebiet des Physical Computing. Aus diesem Grund bietet sich eine technische Unterstützung der SuS an, da diese ein direktes und kompetentes sowie adaptives Feedback ermöglicht (vgl. Spikol et al., 2016). Es existieren bereits diverse Vorarbeiten im Bereich der intelligenten Tutorensysteme. Um diese Systeme geeignet auf den Bereich Physical Computing zu übertragen, sollte eine Kommunikation des ITS mit dem Physical-Computing-Gerät hergestellt und dadurch erhobene Daten von dem Tutorensystem ausgewertet werden. Das erteilte Feedback kann dabei den Problemlöseprozess strukturieren und ebenfalls konkrete Hinweise bezüglich aufgetretener Probleme geben. Eine Vermittlung von Informationen und Grundkenntnissen, z. B. über das Physical-Computing-System und informatische Prinzipien, ist darüber hinaus möglich.

Unterstützung des kollaborativen Lernens

In den durchgeführten Studien wurden Defizite bei der Kommunikation der SuS festgestellt. Insbesondere im Bereich der ITS wird das Feedback zur Unterstützung von Kollaboration bereits untersucht. Deshalb sollten SuS in kollaborativen Lernszenarien technisch unterstützt werden, um effektiv gemeinsam zu arbeiten, wie bereits bei McLaren et al. (2008) und Tsovaltzi et al. (2008) beschrieben. Somit kann das zuvor erwähnte automatisierte Feedback verschiedene Bereiche umfassen, wie z. B. die Unterstützung der Kollaboration durch Empfehlungen bzgl. der Arbeitsteilung und der Anregung zu Diskussionen mit PartnerInnen.

Literaturverzeichnis

Vincent Aleven, Elmar Stahl, Silke Schworm, Frank Fischer, und Raven Wallace. Help seeking and help design in interactive learning environments. *Review of Educational Research*, 73(3):277–320, 2003.

Vincent Aleven, Bruce M. Mclaren, Ido Roll, und Kenneth Koedinger. Toward metacognitive tutoring: A model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education*, 16(2):101–128, 2006.

Louis Alfieri, Patricia J. Brooks, Naomi J. Aldrich, und Harriet R. Tenenbaum. Does discovery-based instruction enhance learning? *Journal of Educational Psychology*, 103 (1):1–18, 2011.

American Association for the Advancement of Science and others. *Benchmarks for scientific literacy*. New York: Oxford University Press, 1993.

John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, und Ray Pelletier. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207, 1995.

Arduino. Arduino Uno Rev3, 2017. URL `https://store.arduino.cc/arduino-uno-rev3`. Zugriffsdatum: 04.08.2017.

Kerstin Arndt. *Experimentierkompetenz erfassen: Analyse von Prozessen und Mustern am Beispiel von Lehramtsstudierenden der Chemie*. Logos Verlag Berlin GmbH, 2016.

Julia Arnold, Kerstin Kremer, und Jürgen Mayer. Understanding students' experiments—what kind of support do they need in inquiry tasks? *International Journal of Science Education*, 36(16):2719–2749, 2014.

Julia Arnold, Kerstin Kremer, und Jürgen Mayer. Scaffolding beim forschenden lernen. *Zeitschrift für Didaktik der Naturwissenschaften*, 23(1):21–37, 2017.

Margarita Azmitia und Ryan Montgomery. Friendship, transactive dialogues, and the development of scientific reasoning. *Social Development*, 2(3):202–221, 1993.

Ryan S. Baker, Albert T Corbett, Kenneth R. Koedinger, und Angela Z. Wagner. Off-task behavior in the cognitive tutor classroom: when students game the system. In

Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI), S. 383–390. ACM, 2004.

Massimo Banzi. *Getting started with Arduino*. O'Reilly Media, Inc., Sebastopol, 2011.

Bildungsserver Berlin-Brandenburg. Rahmenlehrplan für die Jahrgangsstufen 1 –10 der Berliner und Brandenburger Schulen. Teil C: Informatik. Wahlpflichtfach. Jahrgangsstufen 7 –10, 2015. URL `http://bildungsserver.berlin-brandenburg.de/fileadmin/bbb/unterricht/rahmenlehrplaene/Rahmenlehrplanprojekt/amtliche_Fassung/Teil_C_Informatik_2015_11_10_WEB.pdf`. Zugriffsdatum: 31.12.2016.

Paulo Blikstein. Gears of our childhood: constructionist toolkits, robotics, and physical computing, past and future. In *Proceedings of the 12th International Conference on Interaction Design and Children*, S. 173–182. ACM, 2013.

Paulo Blikstein und Uri Wilensky. Bifocal modeling: a framework for combining computer modeling, robotics and real-world sensing. In *Annual Meeting of the American Educational Research Association (AERA 2007), Chicago, USA*. Citeseer, 2007.

Ann L. Brown. Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. *The Journal of the Learning Sciences*, 2 (2):141–178, 1992.

Maja Brückmann und Reinders Duit. Videobasierte analyse unterrichtlicher sachstrukturen. In Dirk Krüger, Ilka Parchmann, und Horst Schecker, Hrsg., *Methoden in der naturwissenschaftsdidaktischen Forschung*, S. 189–201. Springer, Berlin Heidelberg, 2014.

Rodger W. Bybee. Scientific inquiry, student learning, and the science curriculum. *Learning Science and the Science of Learning*, S. 25–35, 2002.

Carnegie Mellon Robotics Academy. Introduction to programming lego mindstorms ev3 teacher's guide, 2014. URL `http://education.rec.ri.cmu.edu/wp-content/uploads/2015/03/EV3-teachers-guideWEB.pdf`. Zugriffsdatum: 02.01.2018.

Zhe Chen und David Klahr. All other things being equal: Acquisition and transfer of the control of variables strategy. *Child Development*, 70(5):1098–1120, 1999.

Clark A. Chinn und William F. Brewer. The role of anomalous data in knowledge acquisition: A theoretical framework and implications for science instruction. *Review of Educational Research*, 63(1):1–49, 1993.

Richard Clark, Paul A. Kirschner, und John Sweller. Putting students on the path to learning: The case for fully guided instruction. *American Educator*, 36(1):6–11, 2012.

Jody Clarke und Chris Dede. Design for scalability: A case study of the river city curriculum. *Journal of Science Education and Technology*, 18(4):353–365, 2009.

David Crismond. Learning and using science ideas when doing investigate-and-redesign tasks: A study of naive, novice, and expert designers doing constrained and scaffolded design work. *Journal of Research in Science Teaching*, 38(7):791–820, 2001.

Jennifer Cross, Emily Hamner, Chris Bartley, und Illah Nourbakhsh. Arts & bots: application and outcomes of a secondary school robotics program. In *Frontiers in Education Conference (FIE)*, S. 1–9. IEEE, 2015.

Jennifer Cross, Emily Hamner, Lauren Zito, und Illah Nourbakhsh. Engineering and computational thinking talent in middle school students: a framework for defining and recognizing student affinities. In *Frontiers in Education Conference (FIE)*, S. 1–9. IEEE, 2016.

Nils Dahlbäck, Arne Jönsson, und Lars Ahrenberg. Wizard of oz studies—why and how. *Knowledge-based systems*, 6(4):258–266, 1993.

Ton De Jong. Technological advances in inquiry learning. *Science*, 312(5773):532–533, 2006.

Ton De Jong und Wouter R. Van Joolingen. Scientific discovery learning with computer simulations of conceptual domains. *Review of Educational Research*, 68(2):179–201, 1998.

Marc J. De Vries. The 'i' in mint: A tale of two translations. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*, WiPSCE '16, S. 1–4. ACM, 2016.

Peter J. Denning. Is computer science science? *Communications of the ACM*, 48(4):27–31, 2005.

Thorsten Dresing und Thorsten Pehl. *Praxisbuch Interview & Transkription. Regelsysteme und Anleitungen für qualitative ForscherInnen.* Dr. Dresing und Pehl GmbH, Marburg, 4 Auflage, 2012.

Fab Lab Berlin. Was ist das Fab Lab Berlinb?, 2016. URL `https://fablab.berlin/de/`. Zugriffsdatum: 24.03.2017.

Uwe Flick. *Qualitative Sozialforschung: Eine Einführung.* Rowohlt, Reinbek, 2007.

Norman M. Fraser und Nigel Gilbert. Simulating speech systems. *Computer Speech & Language*, 5(1):81–99, 1991.

Gesellschaft für Informatik e.V. Grundsätze und Standards für die Informatik in der Schule, 2008. URL `http://www.sn.schule.de/~istandard/docs/bildungsstandards_2008.pdf`. Zugriffsdatum: 02.01.2018.

Richard Gott, Sandra Duggan, Ros Roberts, und Ahmed Hussain. Research into understanding scientific evidence, 2003. URL `http://www.dur.ac.uk/rosalyn.roberts/Evidence/cofev.htm`. Zugriffsdatum: 15.05.2015.

Jakob Gyllenpalm und Per-Olof Wickman. 'Experiments' and the inquiry emphasis conflation in science teacher education. *Science Education*, 95(5):908–926, 2011.

Marcus Hammann und Janina Jördens. Offene Aufgaben codieren. In Dirk Krüger, Ilka Parchmann, und Horst Schecker, Hrsg., *Methoden in der naturwissenschaftsdidaktischen Forschung*, S. 169–178. Springer, Berlin Heidelberg, 2014.

Emily Hamner, Jennifer Cross, Lauren Zito, Debra Bernstein, und Karen Mutch-Jones. Training teachers to integrate engineering into non-technical middle school curriculum. In *Frontiers in Education Conference (FIE)*, S. 1–9. IEEE, 2016.

Friedrich Hesse, Esther Care, Juergen Buder, Kai Sassenberg, und Patrick Griffin. A framework for teachable collaborative problem solving skills. In *Assessment and Teaching of 21st Century Skills*, S. 37–56. Springer, 2015.

Margaret Honey und David E. Kanter. *Design, make, play: Growing the next generation of STEM innovators*. Routledge, New York, 2013.

Dietmar Höttecke. Die Vorstellungen von Schülern und Schülerinnen von der „Natur der Naturwissenschaften". *Zeitschrift für Didaktik der Naturwissenschaften*, 7(1):7–23, 2001.

Johanna Hoysniemi und Janet Read. Wizard of oz studies with children, 2005. URL `https://www.researchgate.net/profile/Janet_Read/publication/228916387_Wizard_of_Oz_Studies_with_Children/links/0deec518618cfd0e52000000.pdf`. Zugriffsdatum: 23.02.2018.

Johanna Höysniemi, Perttu Hämäläinen, und Laura Turkki. Wizard of oz prototyping of computer vision based action games for children. In *Proceedings of the 2004 conference on Interaction design and children: building a community (IDC)*, S. 27–34. ACM, 2004.

Yasmin B. Kafai und Mitchel Resnick. *Constructionism in practice: Designing, thinking, and learning in a digital world*. Routledge, New York, 1996.

Yasmin B. Kafai, Deborah A. Fields, und Kristin Searle. Electronic textiles as disruptive designs: Supporting and challenging maker activities in schools. *Harvard Educational Review*, 84(4):532–556, 2014a.

Yasmin B. Kafai, Eunkyoung Lee, Kristin Searle, Deborah A. Fields, Eliot Kaplan, und Debora Lui. A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles. *ACM Transactions on Computing Education (TOCE)*, 14(1):1, 2014b.

Fatima Kaloti-Hallak, Michal Armoni, und Mordechai Moti Ben-Ari. Students' attitudes and motivation during robotics activities. In *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE)*, S. 102–110. ACM, 2015a.

Fatima Kaloti-Hallak, Michal Armoni, und Ben-Ari Mordechai Moti. The effectiveness of robotics competitions on students' learning of computer science. *Olympiads in Informatics*, 9:89–112, 2015b.

Eva-Sophie Katterfeldt und Nadine Dittert. Ein Framework zur Einordnung programmierbarer Baukästen in interdisziplinäre Bildungskontexte. In I. Diethelm, Hrsg., *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt (INFOS)*, S. 287–290. Köllen Druck+Verlag GmbH, 2017.

Eva-Sophie Katterfeldt, Nadine Dittert, und Heidi Schelhowe. Designing digital fabrication learning environments for bildung: Implications from ten years of physical computing workshops. *International Journal of Child-Computer Interaction*, 5(C):3–10, 2015.

Eva-Sophie Katterfeldt, David Cuartielles, Daniel Spikol, und Nils Ehrenberg. Talkoo: A new paradigm for physical computing at school. In *Proceedings of the The 15th International Conference on Interaction Design and Children (IDC)*, S. 512–517. ACM, 2016.

Brenda Keogh und Stuart Naylor. Concept cartoons, teaching and learning in science: an evaluation. *International Journal of Science Education*, 21(4):431–446, 1999.

Alla Keselman. Supporting inquiry learning by promoting normative understanding of multivariable causality. *Journal of Research in Science Teaching*, 40(9):898–921, 2003.

Diane J. Ketelhut, Brian C. Nelson, Jody Clarke, und Chris Dede. A multi-user virtual environment for building and assessing higher order inquiry skills in science. *British Journal of Educational Technology*, 41(1):56–68, 2010.

Paul A. Kirschner, John Sweller, und Richard E. Clark. Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational psychologist*, 41(2):75–86, 2006.

David Klahr. *Exploring science. The cognition and development of discovery processes.* MIT Press, Cambridge, MA, 2000.

David Klahr und Kevin Dunbar. Dual space search during scientific reasoning. *Cognitive Science*, 12(1):1–48, 1988.

David Klahr und Milena Nigam. The equivalence of learning paths in early science instruction effects of direct instruction and discovery learning. *Psychological Science*, 15 (10):661–667, 2004.

Eckhard Klieme. Was sind Kompetenzen und wie lassen sie sich messen? *Zeitschrift für Pädagogik*, (6):10–13, 2004.

Maria Knobelsdorf, Jonathan Otto, und Sandra Sprenger. A computing education approach for geography students in context of gis. In *Global Engineering Education Conference (EDUCON), 2017 IEEE*, S. 1790–1796. IEEE, 2017.

Martin Kocanda, Bryn M. Wilke, und David S. Ballantine. Using lego mindstorms nxt™ robotics kits as a spectrophotometric instrument. *International Journal on Smart Sensing and Intelligent Systems*, 3(3), 2010.

Kenneth R. Koedinger und Vincent Aleven. Exploring the assistance dilemma in experiments with cognitive tutors. *Educational Psychology Review*, 19(3):239–264, 2007.

Kenneth R. Koedinger und John R. Anderson. Reifying implicit planning in geometry: Guidelines for model-based intelligent. In Susanne P. Lajoie und Sharon J. Derry, Hrsg., *Computers as cognitive tools*, S. 15–45. Routledge, New York, 2013.

Kenneth R. Koedinger, Albert Corbett, et al. Cognitive tutors: Technology bringing learning sciences to the classroom, 2006. URL `http://pact.cs.cmu.edu/pubs/koedingercorbett06.pdf`. Zugriffsdatum: 09.02.2018.

Jenny Koppelt und Rüdiger Tiemann. Computerbasierte erfassung dynamischer problemlösekompetenz. *Chemie-und Physikdidaktik für die Lehramtsausbildung*, 29:265–267, 2009.

Ehud Kroll, Sridhar S. Condoor, und David G. Jansson. *Innovative conceptual design: theory and application of parameter analysis*. Cambridge University Press, 2001.

KMK Kultusministerkonferenz. Sekretariat der Ständigen Konferenz der Kultusminister der Länder in der Bundesrepublik Deutschland (2005): Beschlüsse der Kultusministerkonferenz. Bildungsstandards im Fach Biologie für den Mittleren Schulabschluss, 2005. URL `https://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/2004/2004_12_16-Bildungsstandards-Biologie.pdf`. Zugriffsdatum: 23.02.2018.

J. Richard Landis und Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

Tom Lauwers, Illah Nourbakhsh, und Emily Hamner. Csbots: Design and deployment of a robot designed for the cs1 classroom. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE)*, S. 428–432, New York, NY, USA, 2009. ACM.

Anton E. Lawson. Sound and faulty arguments generated by preservice biology teachers when testing hypotheses involving unobservable entities. *Journal of Research in Science Teaching*, 39(3):237–252, 2002.

LEGO GmbH. LEGO Mindstorms Education EV3, 2017. URL `https://education.lego.com/de-de/product/mindstorms-ev3?CMP=KAC-EDDE17JulEV3AdwordsAdwordsLME`. Zugriffsdatum: 07.08.2017.

Richard Lehrer, Lynn Randle, und Leonard Sancilio. Learning preproof geometry with logo. *Cognition and Instruction*, 6(2):159–184, 1989.

Theodore Lewis. Design and inquiry: Bases for an accommodation between science and technology education in the curriculum? *Journal of Research in Science Teaching*, 43 (3):255–281, 2006.

Marlene Lindner, Sandra Schulz, und Niels Pinkwart. Integration des Erwerbs von Basiskonzepten der Informatik in den mathematisch-naturwissenschaftlichen Unterricht der Sekundarstufe I. In I. Diethelm, Hrsg., *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt (INFOS)*, S. 277–286. Köllen Druck+Verlag GmbH, 2017.

Bernd Mahr. Die Informatik und die Logik der Modelle. *Informatik-Spektrum*, 32(3): 228–249, 2009.

Maker Media, Inc. Maker Faire, 2017. URL `https://maker-faire.de`. Zugriffsdatum: 24.03.2017.

Heidy Maldonado, Scott R. Klemmer, und Roy D. Pea. When is collaborating with friends a good idea? insights from design education. In *Proceedings of the 9th international conference on Computer supported collaborative learning (CSCL)*, S. 227–231. International Society of the Learning Sciences, 2009.

Sarah Manlove, Ard W. Lazonder, und Ton De Jong. Regulative support for collaborative scientific inquiry learning. *Journal of Computer Assisted Learning*, 22(2):87–98, 2006.

Amy M. Masnick und David Klahr. Error matters: An initial exploration of elementary school children's understanding of experimental error. *Journal of Cognition and Development*, 4(1):67–98, 2003.

Jürgen Mayer. Erkenntnisgewinnung als wissenschaftliches Problemlösen. In Dirk Krüger und Helmut Vogt, Hrsg., *Theorien in der biologiedidaktischen Forschung: Ein Handbuch für Lehramtsstudenten und Doktoranten*, S. 177–186. Springer Berlin Heidelberg, 2007.

Philipp Mayring. Qualitative content analysis, 2000. URL `http://nbn-resolving.de/urn:nbn:de:0114-fqs0002204`. Zugriffsdatum: 10.04.2017.

Philipp Mayring. Qualitative Inhaltsanalyse. In Günter Mey und Katja Mruck, Hrsg., *Handbuch qualitative Forschung in der Psychologie*, S. 601–613. VS Verlag für Sozialwissenschaften, Springer Fachmedien Wiesbaden GmbH, 2010.

Jean McKendree. Effective feedback content for tutoring complex skills. *Human-Computer Interaction*, 5(4):381–413, 1990.

Susan McKenney und Thomas C Reeves. *Conducting educational design research*. Routledge, New York, 2012.

Bruce M. McLaren, Nikol Rummel, Niels Pinkwart, Dimitra Tsovaltzi, Andreas Harrer, und Oliver Scheuer. Learning chemistry through collaboration: A wizard-of-oz study of adaptive collaboration support. In *Proceedings of the Workshop on Intelligent Support for Exploratory Environments (ISEE)*. CEUR, 2008.

Bruce M. McLaren, Michael J. Timms, Doug Weihnacht, Daniel Brenner, Kim Luttgen, Andrew Grillo-Hill, und David H. Brown. A web-based system to support inquiry learning-towards determining how much assistance students need. In S. Zvacek, M.T. Restivo, J. Uhomoibhi, und M. Helfert, Hrsg., *Proceedings of the Sixth International Conference on Computer-Supported Education (CSEDU)*, S. 43–52. SCITEPRESS – Science and Technology Publications, 2014.

William I. McWhorter und Brian C. O'Connor. Do lego®mindstorms®motivate students in cs1? In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education (SIGCSE)*, S. 438–442, New York, NY, USA, 2009. ACM.

Patricia L. Merton, Robert K. und Kendall. The focused interview. *American Journal of Sociology*, 51(6):541–557, 1946.

David P. Miller und Cathryne Stein. 'so that's what pi is for!'and other educational epiphanies from hands on robotics. In Allison Druin und James Hendler, Hrsg., *Robots for Kids: Exploring new Technologies for Learning*, S. 219–243. Morgan Kaufmann, San Franciso, 2000.

Andreas Mühling, Alexander Ruf, und Peter Hubwieser. Design and first results of a psychometric test for measuring basic programming abilities. In *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE)*, S. 2–10, New York, NY, USA, 2015. ACM.

National Research Council. *National science education standards*. National Academy Press, Washington, DC, 1996.

Allen Newell und Herbert A Simon. Computer science as empirical inquiry: Symbols and search. *Communications*, 19(3):113–126, 1976.

Next Generation Sciene Standards. HS-LS2-8 Ecosystems: Interactions, Energy, and Dynamics, 2017. URL `http://www.nextgenscience.org/pe/hs-ls2-8-ecosystems-interactions-energy-and-dynamics`. Zugriffsdatum: 18.03.2017.

Kai Niebert und Harald Gropengießer. Leitfadengestützte interviews. In Dirk Krüger, Ilka Parchmann, und Horst Schecker, Hrsg., *Methoden in der naturwissenschaftsdidaktischen Forschung*, S. 121–132. Springer, Berlin Heidelberg, 2014.

OECD. *PISA 2015 Results (Volume V): collaborative problem solving*. OECD Publishing, 2017.

Sandra Y. Okita. The relative merits of transparency: Investigating situations that support the use of robotics in developing student learning adaptability across virtual and physical computing platforms. *British Journal of Educational Technology*, 45(5):844–862, 2014.

Dan O'Sullivan und Tom Igoe. *Physical computing: sensing and controlling the physical world with computers*. Thomson Course Technology Press, Boston, MA, 2004.

Seymour Papert. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc., New York, 1980.

Seymour Papert. Constructionism: A new opportunity for elementary science education, 1989. URL `https://nsf.gov/awardsearch/showAward?AWD_ID=8751190`. Zugriffsdatum: 26.04.2017.

Seymour Papert und Idit Harel. Situating constructionism, 1991. URL `http://namodemello.com.br/pdf/tendencias/situatingconstrutivism.pdf`. Zugriffsdatum: 26.04.2017.

Kerstin Patzwaldt, Meta Kambach, Annette Upmeier zu Belzen, und Rüdiger Tiemann. Experimentierkompetenz erfassen: Zur Entwicklung und Auswertung von Experimentieraufgaben. In Sascha Bernholt, Hrsg., *Naturwissenschaftliche Bildung zwischen Science- und Fachunterricht. Gesellschaft für Didaktik der Chemie und Physik*, Jahrestagung in München 2013, S. 183–185. IPN, 2014.

Mareen Przybylla und Ralf Romeike. Overcoming issues with students' perceptions of informatics in everyday life and education with physical computing-suggestions for the enrichment of computer science classes. In Yasemin Gülbahar und Erinç Karataş, Hrsg., *The International Conference on Informatics in Schools: Situation, Evolution and Perspectives (ISSEP)*, S. 9–20. Springer International Publishing Switzerland 2014, 2014a.

Mareen Przybylla und Ralf Romeike. Physical computing and its scope - towards a constructionist computer science curriculum with physical computing. *Informatics in Education*, 13(2):241 – 254, 2014b.

Mareen Przybylla und Ralf Romeike. Key competences with physical computing. In Torsten Brinda, Nicholas Reynolds, und Ralf Romeike, Hrsg., *KEYCIT 2014: key competencies in informatics and ICT*, S. 216 – 221. Universitätsverlag Potsdam, 2015.

Mareen Przybylla und Ralf Romeike. The nature of physical computing in schools: Findings from three years of practical experience. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research (Koli Calling)*, S. 98–107. ACM, 2017.

Kanjun Qiu, Leah Buechley, Edward Baafi, und Wendy Dubow. A curriculum for teaching computer science through computational textiles. In *Proceedings of the 12th International Conference on Interaction Design and Children (IDC)*, S. 20–27. ACM, 2013.

Mitchel Resnick und Eric Rosenbaum. Designing for tinkerability. In Margaret Honey und David E. Kanter, Hrsg., *Design, make, play: Growing the next generation of STEM innovators*, S. 163–181. Routledge New York, NY, 2013.

Ros Roberts und Richard Gott. A written test for procedural understanding: a way forward for assessment in the uk science curriculum? *Research in Science & Technological Education*, 22(1):5–21, 2004.

Michel Rocard, Peter Csermely, Doris Jorde, Dieter Lenzen, Harriet Walberg-Henriksson, und Valerie Hemmo. Science education now: A renewed pedagogy for the future of europe. *European Commission*, 2007.

Alberto Ruiz, Andrea Bellucci, Paloma Díaz, und Ignacio Aedo. Exploring the use of augmented-reality to support end users in physical computing tasks. In Javed Vassilis Khan, Iris Soute, Antonella De Angeli, Antonio Piccinno, und Andrea Bellucci, Hrsg., *6th International Symposium on End-User Development (IS-EUD)*, S. 72–75, 2017.

Richard M. Ryan und Edward L. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary Educational Psychology*, 25(1):54–67, 2000.

Daniel Salber und Joëlle Coutaz. Applying the wizard of oz technique to the study of multimodal systems. In Juri Bass, Leonard J.and Gornostaev und Claus Unger, Hrsg., *Human-Computer Interaction*, S. 219–230. Springer Berlin Heidelberg, 1993.

William A. Sandoval und Brian J. Reiser. Explanation-driven inquiry: Integrating conceptual and epistemic scaffolds for scientific inquiry. *Science Education*, 88(3):345–372, 2004.

Leona Schauble, Leopold E. Klopfer, und Kalyani Raghavan. Students' transition from an engineering model to a science model of experimentation. *Journal of Research in Science Teaching*, 28(9):859–882, 1991.

Leona Schauble, Robert Glaser, Richard A. Duschl, Sharon Schulze, und Jenny John. Students' understanding of the objectives and procedures of experimentation in the science classroom. *Journal of the Learning Sciences*, 4(2):131–166, 1995.

Florian Schmidt-Weigand, Gudrun Franke-Braun, und Martin Hänze. Erhöhen gestufte Lernhilfen die Effektivität von Lösungsbeispielen? Eine Studie zur kooperativen Bearbeitung von Aufgaben in den Naturwissenschaften. *Unterrichtswissenschaft*, 36(4): 365–384, 2008.

Yannick Schneider und Andreas Mühling. Das Konzept Nature of Computer Science. In I. Diethelm, Hrsg., *Informatische Bildung zum Verstehen und Gestalten der digitalen Welt (INFOS)*, S. 123–126. Köllen Druck+Verlag GmbH, 2017.

Nico Schreiber, Heike Theyßen, und Horst Schecker. Experimentelle Kompetenz messen?! *PhyDid A-Physik und Didaktik in Schule und Hochschule*, 3(8):092–101, 2009.

Sandra Schulz und Niels Pinkwart. Physical computing in stem education. In *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE)*, S. 134–135. ACM, 2015.

Sandra Schulz und Niels Pinkwart. Towards supporting scientific inquiry in computer science education. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education (WiPSCE)*, S. 45–53. ACM, 2016.

Sandra Schulz und Niels Pinkwart. A categorizing taxonomy for occurring problems during robotics activities. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE)*, S. 35–38. ACM, 2017.

Deborah Seehorn, Stephen Carey, Brian Fuschetto, Irene Lee, Daniel Moix, Dianne O'Grady-Cunniff, Barbara B. Owens, Chris Stephenson, und Anita Verno. *CSTA K–12 Computer Science Standards*. Computer Science Teachers Association, New York, NY, USA, 2011.

Sue Sentance und Scarlet Schwiderski-Grosche. Challenge and creativity: Using .net gadgeteer in schools. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education (WiPSCE)*, S. 90–100. ACM, 2012.

Sue Sentance, Jane Waite, Lucy Yeomans, und Emily MacLeod. Teaching with physical computing devices: The bbc micro:bit initiative. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE)*, S. 87–96. ACM, 2017.

SoftBank Robotics. Find out more about nao, 2017. URL `https://www.ald.softbankrobotics.com/en/robots/nao/find-out-more-about-nao`. Zugriffsdatum: 04.08.2017.

Daniel Spikol, Anna Friesel, und Nils Ehrenberg. Supporting robotics education in stem with learning analytics, 2016. URL http://orbit.dtu.dk/ws/files/127819603/4_PELARS_ICR2016_final_all.pdf. Zugriffsdatum: 27.02.2018.

Florence R. Sullivan. Robotics and science literacy: Thinking skills, science process skills and systems understanding. *Journal of Research in Science Teaching*, 45(3):373–394, 2008.

Malcolm Swan. The impact of task-based professional development on teachers' practices and beliefs: a design research study. *Journal of Mathematics Teacher Education*, 10" (4):217–237, 2007.

John Sweller, Jeroen J.G. Van Merrienboer, und Fred G.W.C. Paas. Cognitive architecture and instructional design. *Educational Psychology Review*, 10(3):251–296, 1998.

The LEGO Group. LEGO MINDSTORMS Education EV3; Auszug aus den MINT-Unterrichtsmaterialien mit Schwerpunkt Programmierung/Informatik, 2016a. URL https://le-www-live-s.legocdn.com/sc/media/files/curriculum-grids/ev3-computer-science/ev3-science-curriculum-grid-de-7d65836cef5f1974e52c6a240ec821cf.pdf. Zugriffsdatum: 18.03.2017.

The LEGO Group. LEGO MINDSTORMS Education EV3; Aufgaben für den MINT-Unterricht, 2016b. URL https://le-www-live-s.legocdn.com/sc/media/files/curriculum/le_ev3_aufgaben_fuer_unterricht-30d0cec3ed3ce82fe35811eed0b42a2e.pdf. Zugriffsdatum: 18.03.2017.

Meinald T. Thielsch, Timo Lenzner, und Torsten Melles. Wie gestalte ich gute Items und Interviewfragen? In Meinald T. Thielsch und Torsten Brandenburg, Hrsg., *Praxis der Wirtschaftspsychologie. Bd. II, Themen und Fallbeispiele für Studium und Anwendung*, S. 221–240. Monsenstein u. Vannerdat, 2012.

Marco Thomas. *Informatische Modellbildung: Modellieren von Modellen als ein zentrales Element der Informatik für den allgemeinbildenden Schulunterricht.* Doktorarbeit, Universität Potsdam, 2002.

Dimitra Tsovaltzi, Nikol Rummel, Niels Pinkwart, Andreas Harrer, Oliver Scheuer, Isabel Braun, und Bruce M. McLaren. Cochemex: Supporting conceptual chemistry learning via computer-mediated collaboration scripts. In P. Dillenbourg und M. Specht, Hrsg., *European Conference on Technology Enhanced Learning (EC-TEL)*, S. 437–448. Springer, 2008.

Allen B. Tucker. A model curriculum for k–12 computer science: Final report of the acm k–12 task force curriculum committee. *Learning & Leading with Technology*, 31(7):16 – 20, 2004.

UnternehmerTUM MakerSpace GmbH. MakerSpace, 2016. URL `https://www.maker-space.de`. Zugriffsdatum: 24.03.2017.

Annette Upmeier zu Belzen und Dirk Krüger. Modellkompetenz im Biologieunterricht. *Zeitschrift für Didaktik der Naturwissenschaften*, 16:41–57, 2010.

Alieke M. Van Dijk, Hannie Gijlers, und Armin Weinberger. Scripted collaborative drawing in elementary science education. *Instructional Science*, 42(3):353–372, 2014.

Wouter van Joolingen und Ton de Jong. Exploring a domain with a computer simulation: Traversing variable and relation space with the help of a hypothesis scratchpad. In Douglas Towne, Ton de Jong, und Hans Spada, Hrsg., *Simulation-Based Experiential Learning*, S. 191–206. Springer, 1993.

Wouter R. Van Joolingen, Ton De Jong, und Angelique Dimitrakopoulou. Issues in computer supported inquiry learning in science. *Journal of Computer Assisted Learning*, 23 (2):111–119, 2007.

VERBI Software. Consult. Sozialforschung. GmbH. MaxQDA the Art of Data Analysis, 2017. URL `http://www.maxqda.de`. Zugriffsdatum: 10.04.2017.

Igor M. Verner und David J. Ahlgren. Robot contest as a laboratory for experiential engineering education. *Journal on Educational Resources in Computing*, 4(2):2, 2004.

Maik Walpuski und Elke Sumfleth. Strukturierungshilfen und Feedback zur Unterstützung experimenteller Kleingruppenarbeit im Chemieunterricht. *Zeitschrift für Didaktik der Naturwissenschaften*, 13:181–198, 2007.

Peter C. Wason. On the failure to eliminate hypotheses in a conceptual task. *Quarterly Journal of Experimental Psychology*, 12(3):129–140, 1960.

Scott B. Watson und James E. Marshall. Heterogeneous grouping as an element of cooperative learning in an elementary education science course. *School Science and Mathematics*, 95(8):401–405, 1995.

Barbara Y. White und John R. Frederiksen. Inquiry, modeling, and metacognition: Making science accessible to all students. *Cognition and Instruction*, 16(1):3–118, 1998.

Cameron Wilson, Leigh Ann Sudol, Chris Stephenson, und Mark Stehlik. Running on empty: The failure to teach k-12 computer science in the digital age, 2010. URL `http://runningonempty.acm.org/fullreport2.pdf`. Zugriffsdatum: 21.03.2017.

Nesra Yannier, Kenneth R. Koedinger, und Scott E. Hudson. Tangible collaborative learning with a mixed-reality game: Earthshake. In H. Chad Lane, Kalina Yacef, Jack Mostow, und Philip Pavlik, Hrsg., *International Conference on Artificial Intelligence in Education*, S. 131–140. Springer, 2013.

Corinne Zimmerman. The development of scientific thinking skills in elementary and middle school. *Developmental Review*, 27(2):172–223, 2007.

Corinne Zimmerman und Robert Glaser. Testing positive versus negative claims: A preliminary investigation of the role of cover story on the assessment of experimental design skills. cse technical report. *CSE Technical Report 554*, S. 3 – 18, 2001.

Anhang A

Verwendete Testinstrumente

Messinstrument zur Anwendung von imperativen Kontrollstrukturen

Version 1

Autoren

Andreas Mühling, Alexander Ruf

Lizenz

Für Details zur Erstellung siehe:

Mühling, A., Ruf, A. & Hubwieser, P. (2015). Design and First Results of a Psychometric Test for Measuring Basic Programming Abilities. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (WiPSCE ’15, S. 2–10). New York, NY, USA: ACM. Verfügbar unter http://doi.acm.org/10.1145/2818314.2818320

Auswertung

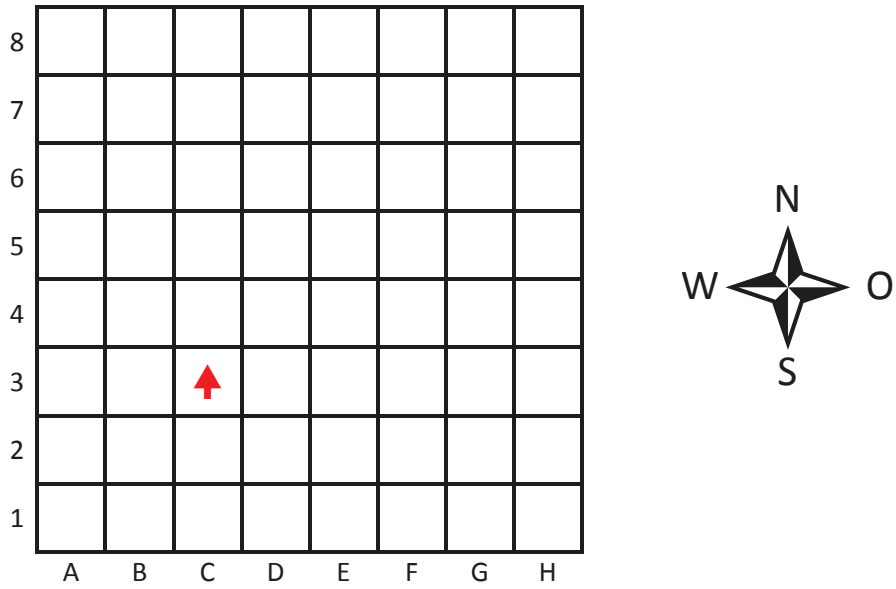
Das Messinstrument ist - nach heutigem Stand der Erkenntnis - Rasch skaliert. Das heißt zur Auswertung genügt es, für jede/n Teilnehmer/-in die Anzahl der richtig gelösten Items zu berücksichtigen. Ein Item gilt als richtig gelöst falls alle Teilantworten des Items richtig sind.

Korrekturschlüssel

Frage 1: D5 / W
Frage 2: D3 / W
Frage 3: D3 / S
Frage 4: G7 / N
Frage 5: C4 / S & C3 / W
Frage 6: F3 / W

Test zu algorithmischen Strukturen

Folgendes Bild zeigt ein Spielbrett mit 8×8 Feldern.



Deine Spielfigur, wir nennen sie „Alex“, steht im Moment im Feld C 3 (siehe roter Pfeil), die Richtung des Pfeiles gibt an, in welche Richtung die Figur ausgerichtet ist (derzeit nach Norden).
Eine Spielfigur kann folgende Anweisungen ausführen:

- `Schritt`
  Die Figur macht einen Schritt in Pfeilrichtung ins benachbarte Feld.
- `Linksdrehen`
  Die Figur dreht sich in Pfeilrichtung gesehen um 90° nach links.
- `Rechtsdrehen`
  Die Figur dreht sich in Pfeilrichtung gesehen um 90° nach rechts.

Alex führt nun folgende Anweisungen der Reihe nach aus:

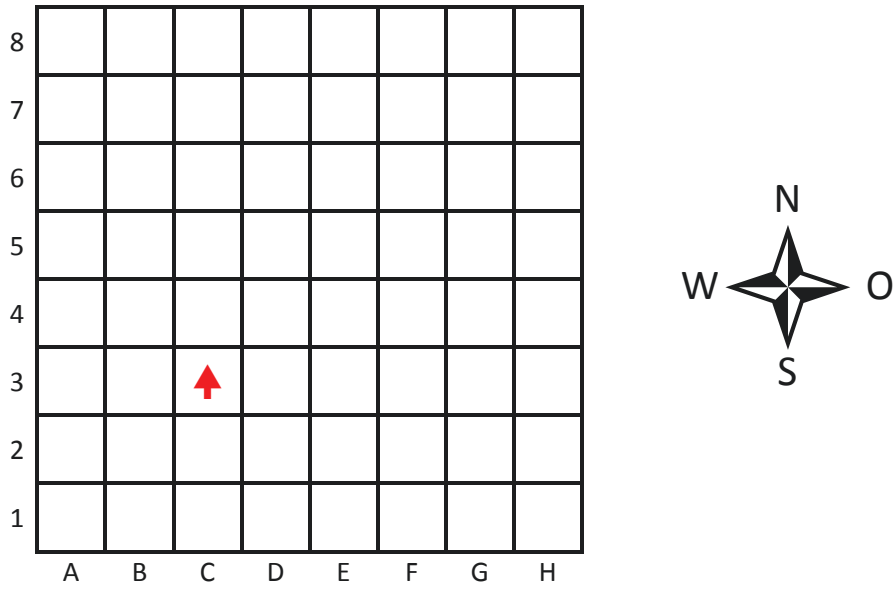
```
Schritt
Schritt
Rechtsdrehen
Schritt
Schritt
Linksdrehen
Linksdrehen
Schritt
```

Gib an, in welchem Feld Alex jetzt steht und in welche Richtung er zeigt.

Feld: _____

Richtung: _____

Alex steht wieder in seiner ursprünglichen Position C 3 und zeigt nach Norden.



Eine Spielfigur kann nun zusätzlich folgende Anweisung ausführen:

- `Wiederhole` *n* `mal`
  `...`
  `EndeWiederhole`
  Die Figur wiederholt die Anweisungen zwischen `Wiederhole` und `EndeWiederhole` *n* mal.

Alex führt nun folgende Anweisungen aus:

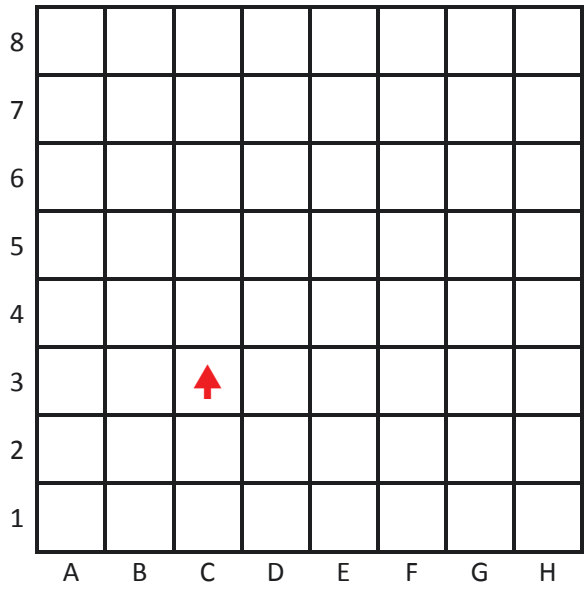
```
Wiederhole 3 mal
    Schritt
    Rechtsdrehen
EndeWiederhole
```

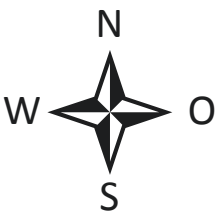
Gib wieder an, in welchem Feld Alex jetzt steht und in welche Richtung er zeigt.

Feld: _____

Richtung: _____

Alex steht wieder in seiner ursprünglichen Position C 3 und zeigt nach Norden.





Alex führt nun folgende Anweisungen aus:

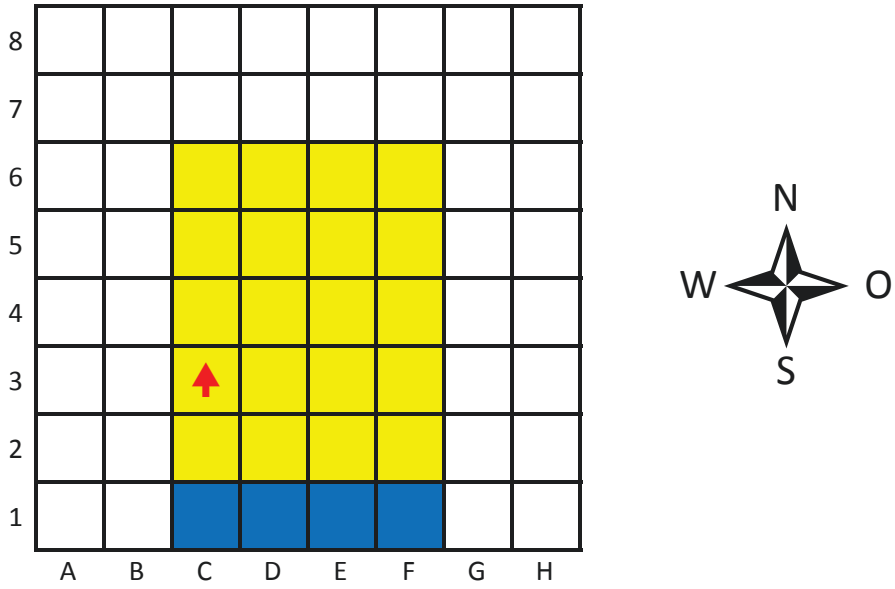
```
Wiederhole 3 mal
    Wiederhole 2 mal
        Schritt
        Rechtsdrehen
    EndeWiederhole
    Schritt
EndeWiederhole
```

Gib wieder an, in welchem Feld Alex jetzt steht und in welche Richtung er zeigt.

Feld: _____

Richtung: _____

Folgendes Bild zeigt wieder das Spielbrett mit 8×8 Feldern, allerdings können die Felder nun auch gelb oder blau gefärbt sein.



Eine Spielfigur kann nun zusätzlich folgende Anweisung ausführen:

- `Wiederhole solange` *Bedingung*
  `...`
  `EndeWiederhole`
  Die Figur wiederholt die Anweisungen zwischen `Wiederhole` und `EndeWiederhole` solange, bis die *Bedingung* `falsch` ist.

Als *Bedingung* stehen folgende zwei Möglichkeiten zur Verfügung:

- `IstGelb` bzw. `IstBlau`
  Liefert `wahr`, falls das Feld, **auf dem die Figur steht**, gelb bzw. blau gefärbt ist, ansonsten `falsch`.

Alex führt nun folgende Anweisungen aus:

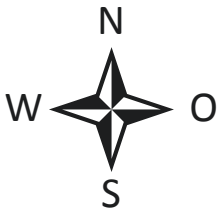
```
Wiederhole solange IstGelb
    Rechtsdrehen
    Schritt
    Linksdrehen
    Schritt
EndeWiederhole
```

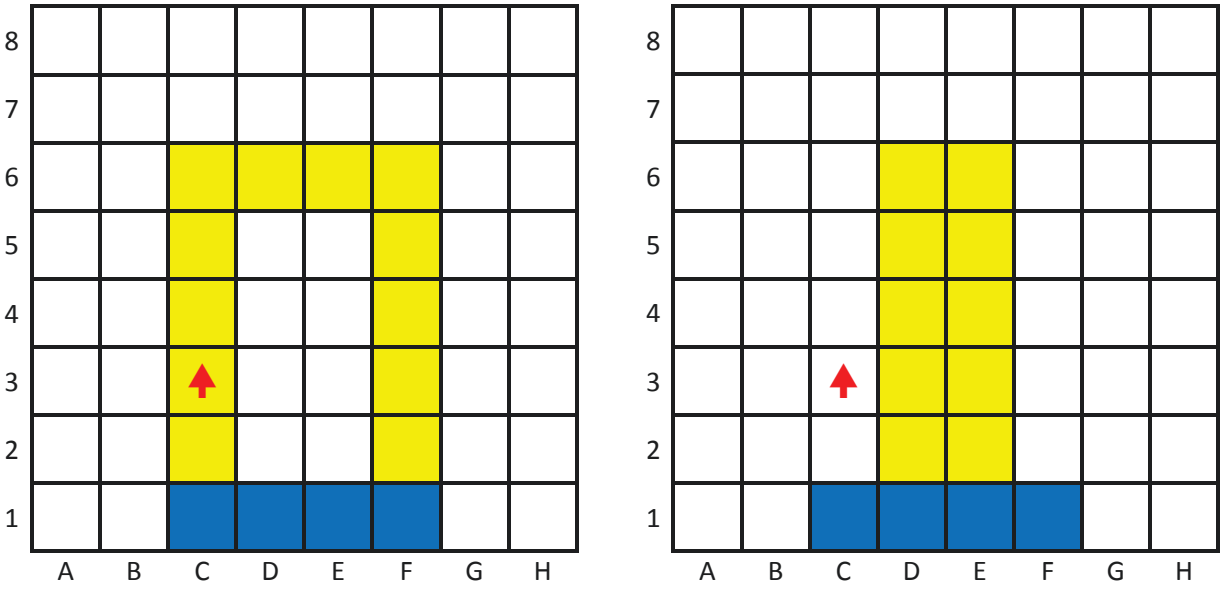
Gib wieder an, in welchem Feld Alex jetzt steht und in welche Richtung er zeigt.

Feld: _____

Richtung: _____

Alex steht in beiden der folgenden Spielfelder in seiner ursprünglichen Position C 3 und zeigt nach Norden.





Eine Spielfigur kann nun zusätzlich folgende Anweisungen ausführen:

- `Falls` *Bedingung* `Dann`
  `...`
  `EndeFalls`
  Die Figur führt die Anweisungen zwischen `Dann` und `EndeFalls` nur aus, falls die *Bedingung* `wahr` ist, ansonsten werden diese Anweisungen nicht ausgeführt.

Alex führt nun folgende Anweisungen aus, **<u>im linken genauso wie im rechten</u>** Spielfeld:

```
Falls IstGelb Dann
    Schritt
    Linksdrehen
EndeFalls
Linksdrehen
```

Gib **<u>für beide</u>** Spielfelder an, in welchem Feld Alex jetzt steht und in welche Richtung er zeigt.

<u>Linkes Spielfeld:</u>

Feld: _____

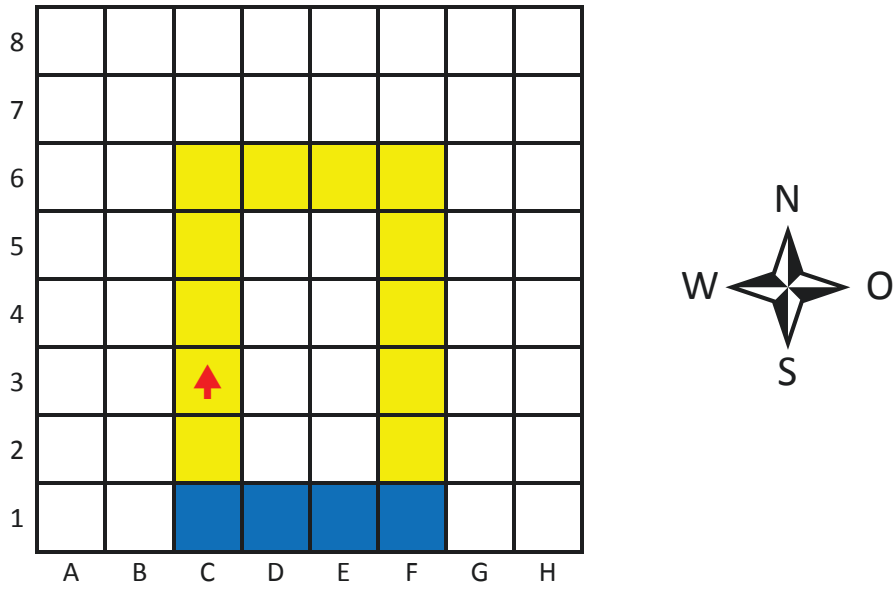
Richtung: _____

<u>Rechtes Spielfeld:</u>

Feld: _____

Richtung: _____

Alex steht wieder in seiner ursprünglichen Position C 3 und zeigt nach Norden.



Die `Falls`-Anweisung einer Spielfigur wird nun folgendermaßen erweitert:

- `Falls` *Bedingung* `Dann`
  ```
      ...
  Sonst
      ...
  EndeFalls
  ```
  Die Figur führt die Anweisungen zwischen `Dann` und `Sonst` aus, wenn die *Bedingung* `wahr` ist, und die Anweisungen zwischen `Sonst` und `EndeFalls`, wenn die *Bedingung* `falsch` ist.

Alex führt nun folgende Anweisungen aus:

```
Wiederhole 4 mal
    Schritt
    Falls IstGelb Dann
        Rechtsdrehen
    Sonst
        Schritt
    EndeFalls
EndeWiederhole
```

Gib wieder an, in welchem Feld Alex jetzt steht und in welche Richtung er zeigt.

Feld: _____

Richtung: _____

Fragebogen zur Evaluation von Problemen in der Robotik/Physical Computing

Bei der Arbeit mit Robotern und anderen Physical Computing Geräten gibt es verschiedene Fehlerquellen und Probleme, die uns in anderen Bereichen der Informatik seltener begegnen. Im Folgenden sollst du die Häufigkeit von auftretenden Fehlerquellen einschätzen, sowie konkrete Aktionen und Probleme in zugehörige Kategorien einordnen. Dafür betrachten wir einen Roboter, der mit einer Solarzelle, einem Temperatursensor, einem Bewegungssensor, einem Lichtsensor, einer Stifthalterung und einem Display ausgestattet ist. Die aufgenommenen Daten werden von den Sensoren selbst nicht weiterverarbeitet. Der Roboter wird über einen Knopf an- und ausgeschaltet.

Gib bitte zuvor an, wie viel Vorerfahrung du auf folgenden Gebieten hast. Kreuze an!

Einschätzung deiner Vorerfahrungen	sehr wenig 1	2	3	4	5	sehr viel 6
Informatik allgemein.						
Programmierung.						
Arbeit mit LEGO-Robotern.						
Arbeit mit anderen Physical Computing Geräten, wie z.B. Arduino oder Raspberry Pi.						

Gibt bitte an, ich welcher Klassenstufe du dich aktuell befindest: _____

Aufgabe 1

Wie häufig meinst du, kommen die folgenden Fehlerquellen beim Lösen von Aufgaben mit Robotern vor? Kreuze an!

Eine häufige Fehlerquelle ist:	stimmt gar nicht 1	2	3	4	5	stimmt sehr 6
…Software (z.B. die selbst geschriebene auf dem Roboter oder die Programmierumgebung).						
…Hardware (z.B. die Verkabelung am Roboter und die Position von Sensoren).						
…Umgebungseinflüsse (z.B. Veränderungen von Umweltfaktoren oder das Eingreifen des Menschen in das laufende System).						

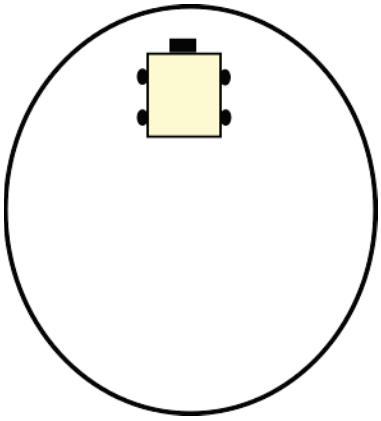
Aufgabe 2

Unten sind verschiedene Aktionen angegeben. Entscheide, ob es sich dabei vorrangig um eine Eingabe von Informationen (Input), eine Verarbeitung von Informationen (Processing) oder eine Ausgabe von Informationen (Output) handelt. Unterscheide beim Input, ob dieser analog oder digital ist. Es ist nur ein Kreuz pro Aktion möglich. Kreuze an!

Aktion	Input		Processing	Output
	analog	digital		
a. Der Roboter steht in einem Raum und der Bewegungssensor am Roboter wird ausgelöst.				
b. Ein Roboter mit Stifthalterung setzt den Stift auf dem Untergrund auf.				
c. Der Roboter wartet, bis eine spezielle Aktion passiert.				
d. Der Roboter fährt geradeaus.				
e. Der Roboter zeichnet einen Kreis auf den Boden auf.				
f. Der Roboter berechnet den kürzesten Weg zur Lichtquelle.				
g. Das Display am Roboter leuchtet auf.				
h. Die Sonne scheint auf den Roboter.				
i. Der Roboter wird angeschaltet.				
j. Die eingelesenen Werte des Temperatursensors werden in Grad Celsius umgerechnet.				
k. Der Roboter überprüft die Höhe der Stifthalterung.				
l. Aus den eingelesenen Sensordaten wird die Durchschnittstemperatur ermittelt.				

Begründe kurz deine Entscheidung:

Aufgabe 3



Ein Roboter mit einem nach unter gerichteten Lichtsensor soll sich ausschließlich innerhalb des schwarzen Kreises bewegen und nicht über die schwarze Linie fahren (siehe Bild). Bei der Bearbeitung der Aufgabe haben Lars und Lara das Problem, dass der Roboter einfach über die Linie hinweg fährt.

Beschreibe für jede Fehlerquelle drei Möglichkeiten, die zu diesem Verhalten des Roboters geführt haben können!

Software:

- ……
- ……
- ……

Hardware:

- ……
- ……
- ……

Umgebung:

- ……
- ……
- ……

Anhang B

Verwendete Hilfestellungen

B.1 Hilfestellung *4-Phasen analog*

Arbeitsblatt 3

Aufgabe: Programmiert den Roboter so, dass er unter Benutzung des Ultraschallsensors um eine Kiste fahren kann.

1. Wie muss sich der Roboter verhalten, um die Aufgabe zu lösen? Wann ist die Aufgabe gelöst?	2. Was soll euer Programm machen? Wie verändert es sich?
3. Was könnt ihr bei dem Ausführen des Programms beobachten?	4. Was müsst ihr noch verändern, damit die Aufgabe gelöst werden kann?

B.2 Hilfestellung *Evaluationsphase digital*

Roboter-Workshop

Ihr werdet bei der Lösung der Aufgabe immer wieder die folgenden 4 Phasen durchlaufen. Dabei beschreibt ihr bitte nach jedem Durchlauf die Phase Durchführung (gelb in der Tabelle) und Auswertung (grün in der Tabelle).

1. Vermutung anstellen wie die Aufgabe gelöst werden kann
2. Programmieren (am PC)
3. Durchführung (Programm auf dem Roboter testen)
4. Auswertung (Lösung bewerten und Fehlerquellen ermitteln für den nächsten Durchlauf)

Nr.	Beschreibung der Bewegung des Roboters bei der Ausführung des Programms	Mögliche Fehlerquellen			Welche Fehlerquelle ist am wahrschein-lichsten?
		Euer Programm (Software)	Bauteile am Roboter (Hardware)	Umgebung um Roboter (Umwelt)	
Bsp.	Aufgabe Linie folgen: Roboter fährt 1 Sekunde auf der Linie und dreht sich dann im Kreis.	Der Roboter fährt zu schnell und die Linie wird deswegen nicht erkannt	Lichtsensor kaputt	Lichtverhältnisse im Raum wechseln ständig oder zu unterschiedlich	Software
1					
2					
3					

B.3 Hilfestellung *Problemtaxonomie gestuft*

Hilfestellungen zur Problemlösung in den Interventionsstunden der Schülergesellschaft SoSe 2017

Ebenen der Hilfestellung

1. Stufe

Eingrenzung in 3 Hauptfehlerquellen bzw. ihre Überschneidungen (HW/SW, usw.).

2. Stufe

Beschreibung der Unterkategorie der Fehlerquellen (SW: Programmcode, SW auf dem Roboter selbst oder Umgang mit der Programmieroberfläche).

3. Stufe (wird ggf. übersprungen)

Genauere Einschränkung des Fehlerbereichs ohne explizite Nennung des Fehlers (als Aussage formuliert).

4. Stufe

Klare Benennung des aufgetretenen Fehlers und wie dieser behoben werden kann.

Beispiel: Roboter fährt über eine am Boden gezeichnete Linie hinaus: Wahrscheinliches Problem, dass Schwellwert falsch gesetzt wurde.

1. Stufe: Problem liegt in SW.

2. Stufe: Problem liegt in dem Programmcode.

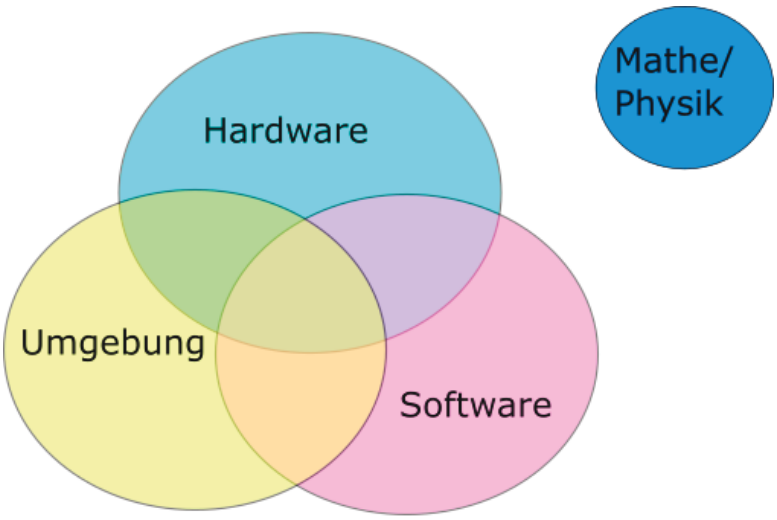
3. Stufe: Eine Variable hat einen falschen Wert zugewiesen bekommen / In dem Schalter, wo der Sensorwert abgefragt wird, ist ein Fehler.

4. Stufe: Der Schwellwert muss niedriger gesetzt werden, z.B. auf den Wert 30.

Anschließendes Interview

- Gruppeninterview (jeweils 2 Gruppen (ca. 4 Personen – zugehörig zu den Wizards)
- 16 SuS → 4 Gruppen → ca. 2 Minuten pro Interview, also in den letzten 10 min der SG
- Fragen:
    1. Welche Probleme hattet ihr beim Lösen der Aufgaben? (Hilfe: Ihr könnt euch dabei auf die Hauptfehlerquellen beziehen.)
    2. War die Hilfestellung eurer BetreuerIn hilfreich?
    3. Konntet ihr auftretende Fehler besser beheben oder verstehen? Wenn ja, wieso?

Taxonomie von auftretenden Problemen



Sub-Kategorien:

Hardware:

- Konstruktion
- Hardwarebestandteile funktionieren nicht
- Hardwarebestandteile funktionieren fehlerhaft

Software:

- Programmcode (selbst geschrieben)
- Programmierumgebung auf dem PC
- Software auf dem Roboter

Umgebung:

- Natürliche Umgebungseinflüsse (Lichtbedingungen)
- Menschliches Eingreifen und Verändern der Umgebung

Mathe/Physik:

- Verwendung von Operatoren
- Physikalische Funktionsweise von Sensoren
- Aufbau eines physikalischen Experiments
- Bestimmung von Schwellwerten

Anhang C

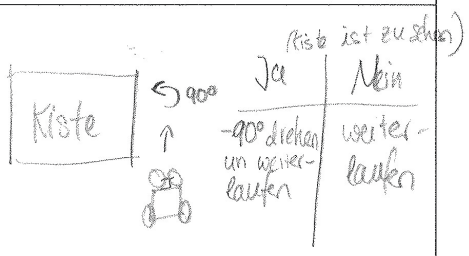
Exemplarische Antworten der SuS unter Verwendung der Hilfestellungen

C.1 Hilfestellung *4-Phasen analog* Beispiel I

Roboter-Workshop 01.02.2016

Arbeitsblatt 3

Aufgabe: Programmiert den Roboter so, dass er unter Benutzung des Ultraschallsensors um eine Kiste fahren kann.

1. Wie muss sich der Roboter verhalten, um die Aufgabe zu lösen? Wann ist die Aufgabe gelöst?	2. Was soll euer Programm machen? Wie verändert es sich?
- er muss die Kiste wahrnehmen - und solange vorwärts fahren, bis der er die Kiste nicht mehr wahrnimmt - kann aber erst drehen, wenn er weit genug von der Kiste weg ist	
3. Was könnt ihr bei dem Ausführen des Programms beobachten?	4. Was müsst ihr noch verändern, damit die Aufgabe gelöst werden kann?

4/6

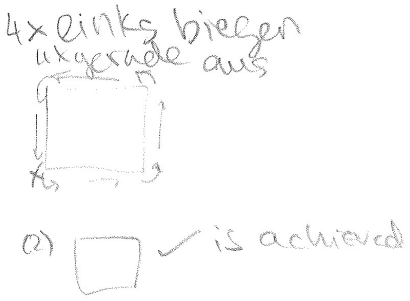
6 = 360°
1 < 60
4 · 4 = 240°

C.2 Hilfestellung *4-Phasen analog* Beispiel II

Roboter-Workshop 01.02.2016

Arbeitsblatt 3

Aufgabe: Programmiert den Roboter so, dass er unter Benutzung des Ultraschallsensors um eine Kiste fahren kann.

1. Wie muss sich der Roboter verhalten, um die Aufgabe zu lösen? Wann ist die Aufgabe gelöst?	2. Was soll euer Programm machen? Wie verändert es sich?
	
3. Was könnt ihr bei dem Ausführen des Programms beobachten?	4. Was müsst ihr noch verändern, damit die Aufgabe gelöst werden kann?

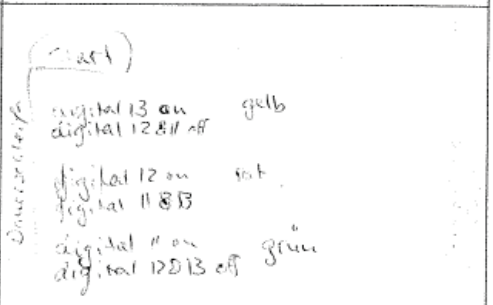
C.3 Hilfestellung *4-Phasen analog* Beispiel III

Aufgabe 1:

Lest den Photowiderstand aus und programmiert als Output drei verschiedene Zustände:

1. Der Photowiderstand wird abgedeckt - eine rote LED leuchtet auf.
2. Der Photowiderstand ist senkrecht aufgerichtet im Raum - eine gelbe LED leuchtet auf.
3. Der Photowiderstand wird mit einer Lampe angestrahlt - eine grüne LED leuchtet auf.

Macht Notizen zu eurem Lösungsweg!

1. Formuliert einen Lösungsweg. Wann ist das Problem gelöst?	2. Wie muss das Programm dafür aussehen? Welche Veränderung nehmt ihr im Programm vor?
	
3. Startet das Programm! Was beobachtet ihr dabei?	4. Was müsst ihr noch verändern, um das Problem zu lösen? Welche Probleme sind aufgetreten?

C.4 Hilfestellung *Evaluationsphase digital* Beispiel I

Roboter-Workshop

Ihr werdet bei der Lösung der Aufgabe immer wieder die folgenden 4 Phasen durchlaufen. Dabei beschreibt ihr bitte nach jedem Durchlauf die Phase Durchführung (gelb in der Tabelle) und Auswertung (grün in der Tabelle).

1. Vermutung anstellen wie die Aufgabe gelöst werden kann
2. Programmieren (am PC)
3. Durchführung (Programm auf dem Roboter testen)
4. Auswertung (Lösung bewerten und Fehlerquellen ermitteln für den nächsten Durchlauf)

Nr.	Beschreibung der Bewegung des Roboters bei der Ausführung des Programms	Mögliche Fehlerquellen			Welche Fehlerquelle ist am wahrschein-lichsten?
		Euer Programm (Software)	Bauteile am Roboter (Hardware)	Umgebung um Roboter (Umwelt)	
Bsp.	Aufgabe Linie folgen: Roboter fährt 1 Sekunde auf der Linie und dreht sich dann im Kreis.	Der Roboter fährt zu schnell und die Linie wird deswegen nicht erkannt	Lichtsensor kaputt	Lichtverhältnisse im Raum wechseln ständig oder zu unterschiedlich	Software
1	Roboter fährt gegen die Kiste	Werte stimmen nicht	Ultraschallsensor hat Kiste nicht wahrgenommen		Software
2	Roboter fährt zwar um die Kiste fährt aber teilweise gegen die Kiste	Roboter lenkt nicht weit genug ein, Distanz für Ultraschall muss höher sein	Ultraschallsensor hat Kiste nicht wahrgenommen		Software
3	Ausweichbewegung zu groß	Wert für Motor zu groß			Software

4	Nachdem der Roboter um die Kurve gefahren ist prüft er sofort, ob links die Kiste ist und da ist sie natürlich nicht	Nach der Kurve wird sofort überprüft			Software
5	Roboter fährt um die Kiste, indem er immer nach links fährt und sich so Stück für Stück an der Kiste langschiebt Manchmal hat der Ultraschall die Kiste eigentlich hätte sehen müssen hat es aber nicht	Werte nicht richtig	Ultraschallsensor		Software
6	Roboter driftet sehr weit nach rechts ab, obwohl die Bewegung nach links stärker ist als nach rechts		Motor unterschiedlich in Vorwärts- und Rückwärtsbewegung	Boden nicht glatt und nicht gerade	Hardware → Software dem entsprechend anpassen
7	Roboter dreht sich direkt am Anfang Ausweichbewegungen zu groß	Werte zu groß	Motor unterschiedlich (siehe oben)	Boden nicht glatt und nicht gerade	Software
8	Roboter dreht sich direkt am Anfang				Hand nicht vor den Sensor halten
9	Drehungen ziemlich klein, fährt gegen die Wand bevor er sie wahrnimmt	Drehungswert erhöhen, Abstandswert erhöhen			Software

Anhang D

Testdaten der Wizard-of-Oz-Studien

Tabelle D.1: Ergebnisse Pretest der Wizard-of-Oz-Studie I, Basis-Programmierkompetenz-Test

ID	A1	A2	A3	A4	A5_1	A5_2	A6	$\sum$ (von 7)
WoZ_1_K_PRE-01	1	1	1	1	1	1	1	7
WoZ_1_K_PRE-02	1	0	0	0	0	0	1	2
WoZ_1_K_PRE-03	1	1	1	1	1	1	1	7
WoZ_1_K_PRE-04	0	1	1	1	1	1	1	6
WoZ_1_K_PRE-05	1	1	1	1	0	1	0	5
WoZ_1_K_PRE-06	0	1	0	0	0	0	0	1
WoZ_1_K_PRE-07	1	1	1	1	1	1	1	7
WoZ_1_K_PRE-08	1	1	0	0	0	0	0	2
WoZ_1_K_PRE-09	1	1	0	0	0	0	0	2
WoZ_1_K_PRE-10	1	1	0	0	1	1	0	4
WoZ_1_K_PRE-11	1	0	0	0	1	1	0	3
WoZ_1_K_PRE-12	0	0	0	0	0	0	0	0
WoZ_1_K_PRE-13	1	1	1	0	1	1	0	5
WoZ_1_K_PRE-14	1	1	1	0	1	1	0	5

Tabelle D.2: Auswertung Pretest der Wizard-of-Oz-Studie I, Basis-Programmierkompetenz-Test

Antwortwahrscheinlichkeit in %	Mittelwert	Median	Min	Max
57	4	4,5	0	7

Tabelle D.3: Ergebnisse Posttest der Wizard-of-Oz-Studie I, Basis-Programmierkompetenz-Test

ID	A1	A2	A3	A4	A5_1	A5_2	A6	$\sum$ (von 7)
WoZ_1_K_POS-01	1	1	1	1	0	1	1	6
WoZ_1_K_POS-02	1	0	0	1	0	1	0	3
WoZ_1_K_POS-03	1	1	1	1	1	1	1	7
WoZ_1_K_POS-04	1	0	0	0	1	0	0	2
WoZ_1_K_POS-05	1	1	1	1	0	1	1	6
WoZ_1_K_POS-06	1	1	0	0	0	1	1	4
WoZ_1_K_POS-07	1	1	1	0	0	1	0	4
WoZ_1_K_POS-08	1	1	1	1	1	1	0	6
WoZ_1_K_POS-09	1	1	0	0	0	1	1	4
WoZ_1_K_POS-10	0	1	0	0	0	0	0	1
WoZ_1_K_POS-11	1	1	0	0	1	1	0	4
WoZ_1_K_POS-12	1	1	1	1	1	1	1	7
WoZ_1_K_POS-13	1	1	0	1	0	1	1	5
WoZ_1_K_POS-14	1	1	1	0	0	1	1	5

Tabelle D.4: Auswertung Posttest der Wizard-of-Oz-Studie I, Basis-Programmierkompetenz-Test

Antwortwahrscheinlichkeit in %	Mittelwert	Median	Min	Max
65	4,6	5	1	7

Tabelle D.5: Ergebnisse Pretest der Wizard-of-Oz-Studie II, Basis-Programmierkompetenz-Test

ID	A1	A2	A3	A4	A5_1	A5_2	A6	$\sum$ (von 7)
WoZ_2_K_PRE-01	1	0	1	1	0	1	1	5
WoZ_2_K_PRE-02	1	1	1	1	1	1	1	7
WoZ_2_K_PRE-03	1	1	1	1	1	1	1	7
WoZ_2_K_PRE-04	1	1	1	0	0	1	1	5
WoZ_2_K_PRE-05	0	0	0	0	1	1	0	2
WoZ_2_K_PRE-06	1	1	1	0	0	1	1	5

Tabelle D.6: Auswertung Pretest der Wizard-of-Oz-Studie II, Basis-Programmierkompetenz-Test

Antwortwahrscheinlichkeit in %	Mittelwert	Median	Min	Max
73	5,2	5	2	7

Tabelle D.7: Ergebnisse Posttest der Wizard-of-Oz-Studie II, Basis-Programmierkompetenz-Test

ID	A1	A2	A3	A4	A5_1	A5_2	A6	$\sum$ (von 7)
WoZ_2_K_POS-01	1	1	1	1	1	1	1	7
WoZ_2_K_POS-02	1	1	1	0	1	1	1	6
WoZ_2_K_POS-03	1	1	0	0	0	1	1	4
WoZ_2_K_POS-04	1	1	1	1	1	1	0	6
WoZ_2_K_POS-05	1	1	0	0	1	1	0	4
WoZ_2_K_POS-06	1	0	0	1	1	1	0	4

Tabelle D.8: Auswertung Posttest der Wizard-of-Oz-Studie II, Basis-Programmierkompetenz-Test

Antwortwahrscheinlichkeit in %	Mittelwert	Median	Min	Max
73	5,2	5	4	7

Tabelle D.9: Ergebnisse Pretest Wizard-of-Oz-Studie I, Physical-Computing-Test

Items	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_10	P_11	P_12	P_13	P_14	MW	MW_Kategorie (gesamt)
Aufgabe 1																
Informatik allgemein	3	5	5	4	5	5	4	4	2	3	4	4	4	4	4,00	
Programmierung	4	3	6	4	4	5	5	4	2	4	2	3	2	4	3,71	
LEGO-Roboter	4	3	6	3	2	5	2	4	4	5	2	4	2	4	3,57	
PhC-Geräte	1	1	6	3	1	1	4	1		3	2	2	1	2	2,15	
Software	2	6	5	2	5	6	4	2	2	5	5	4	5		4,08	
Hardware	4	3	3	2	3	4	2	3	1	4	4	2	5		3,08	
Umgebung	5	1	2	5	4	3	4	2	3	4	4	3	6		3,54	
Aufgabe 2																
Input 1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	0,79	0,79
Output 1	0	1	1	1	0	1	1	0	0	0	0	0	1	1	0,50	0,70
Processing 1	0	0	1	1	1	1	1	0	1	0	0	1	1	1	0,64	0,88
Output 2	0	1	1	1	0	1	1	0	0	1	0	0	1	1	0,57	
Output 3	1	1	1	1	0	1	1	0	1	1	0	1	1	1	0,79	
Processing 2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1,00	
Output 4	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0,93	
Input 2	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0,93	
Input 3	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0,93	
Processing 3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1,00	
Input 4	0	0	1	1	0	0	1	0	0	0	1	1	1	1	0,50	
Processing 4	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0,86	

Aufgabe 3																
Software 1	1	0	1	0	0	0	1	0	1	1	0	0	0	1	0,43	0,40
Software 2	1	0	1	0	1	0	1	0	0	1	0	1	0	0	0,43	
Software 3	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0,36	
Hardware 1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	0,86	0,60
Hardware 2	1	0	1	1	0	1	1	1	1	0	0	1	0	1	0,64	
Hardware 3	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0,29	
Umgebung 1	0	1	1	1	1	1	1	0	1	0	0	1	1	1	0,71	0,50
Umgebung 2	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0,29	
Umgebung 3	1	0	1	1	0	1	0	0	1	0	0	0	1	1	0,50	

Tabelle D.10: Ergebnisse Posttest Wizard-of-Oz-Studie I, Physical-Computing-Test

Items	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_10	P_11	P_12	P_13	P_14	MW	MW_Kategorie (gesamt)
Aufgabe 1																
Informatik allgemein	4	4	4	3	5	5	5	5	4	4	4	5	3	5	4,29	
Programmierung	3	5	4	4	6	6	5	5	5	3	3	5	3	3	4,29	
LEGO-Roboter	3	5	5	3	5	3	4	6	3	4	4	5	4	3	4,07	
PhC-Geräte	1	2	4	5	5	2		2	5	1	2	2	1	4	2,77	
Software	6	5	5	5	5	3	5	2	6	4	6	5	4	4	4,64	
Hardware	3	4	3	3	2	4	2	2	4	2	2	3	3	2	2,79	
Umgebung	4	5	2	2	4	5	5	3	4	4	4	2	6	3	3,79	
Aufgabe 2																
Input 1	1	0	1	0	1	1	1	1	1	0	1	1	1	1	0,79	0,69
Output 1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0,93	0,89
Processing 1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0,86	0,88
Output 2	0	1	1	1	1	1	1	1	1	0	1	1	1	1	0,86	
Output 3	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0,93	
Processing 2	1	0	1	1	1	1	1	1	1	0	1	1	1	1	0,86	
Output 4	1	1	1	1	1	1	0	1	1	0	1	1	1	1	0,86	
Input 2	1	1	1	1	1	1	0	1	1	0	1	1	1	1	0,86	
Input 3	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0,86	
Processing 3	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0,93	
Input 4	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0,29	
Processing 4	1	0	1	1	1	1	1	1	1	0	1	1	1	1	0,86	

Aufgabe 3																
Software 1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	0,57	0,62
Software 2	0	1	1	1	1	1	1	1	1	1	0	1	0	1	0,79	
Software 3	1	1	1	0	1	0	1	0	0	0	0	1	1	0	0,50	
Hardware 1	1	1	1	1	1	0	0	1	1	1	1	0	1	1	0,79	0,69
Hardware 2	1	0	1	1	1	1	0	1	1	1	1	0	0	1	0,71	
Hardware 3	0	1	1	0	1	1	1	0	1	0	0	0	1	1	0,57	
Umgebung 1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	0,79	0,66
Umgebung 2	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0,71	
Umgebung 3	1	0	1	0	1	0	0	0	1	0	1	1	0	1	0,50	

Tabelle D.11: Ergebnisse Pretest Wizard-of-Oz-Studie II, Physical-Computing-Test

Items	P_1	P_2	P_3	P_4	P_5	P_6	MW	MW_Kategorie (gesamt)
Aufgabe 1								
Informatik allgemein	3	3	4	4	5	1	3,33	3,50
Programmierung	3	3	4	4	6	1	3,50	3,50
LEGO-Roboter	2	1	3	1	4	2	2,17	2,00
PhC-Geräte	1	3	2	1	1	1	1,50	1,00
Software	1	4	6	3	5	4	3,83	4,00
Hardware	4	6	5	3	5	1	4,00	4,50
Umgebung	4	5	4	5	2	2	3,67	4,00
Aufgabe 2								
Input 1	1	1	1	1	1	0	0,83	0,88
Output 1	0	1	1	1	1	1	0,83	0,88
Processing 1	0	1	1	0	1	0	0,50	0,79
Output 2	1	1	1	1	1	0	0,83	
Output 3	1	1	1	1	1	0	0,83	
Processing 2	1	1	1	1	1	0	0,83	
Output 4	1	1	1	1	1	1	1,00	
Input 2	1	1	1	1	1	1	1,00	
Input 3	1	1	1	1	1	1	1,00	
Processing 3	1	1	1	1	1	0	0,83	
Input 4	0	1	1	1	1	0	0,67	
Processing 4	1	1	1	1	1	1	1,00	
Aufgabe 3								
Software 1	1	1	1	1	1	0	0,83	0,67
Software 2	1	1	0	1	1	0	0,67	
Software 3	0	1	0	0	1	1	0,50	
Hardware 1	1	0	1	1	1	0	0,67	0,67
Hardware 2	0	1	1	1	1	0	0,67	
Hardware 3	0	0	1	1	1	1	0,67	
Umgebung 1	1	1	1	1	1	1	1,00	0,67
Umgebung 2	0	0	1	1	1	0	0,50	
Umgebung 3	0	0	0	0	0	1	0,17	

Tabelle D.12: Ergebnisse Posttest Wizard-of-Oz-Studie II, Physical-Computing-Test

Items	P_1	P_2	P_3	P_4	P_5	P_6	MW	MW_Kategorie (gesamt)
Aufgabe 1								
Informatik allgemein	4	1	3	4	5	5	3,67	4,00
Programmierung	3	1	3	4	6	5	3,67	3,50
LEGO-Roboter	1	2	3	3	4	3	2,67	3,00
PhC-Geräte	4	1	1	2	1	2	1,83	1,50
Software	5	2	3	6	6	4	4,33	4,50
Hardware	3	5	6	5	3	5	4,50	5,00
Umgebung	4	3	5	4	5	4	4,17	4,00
Aufgabe 2								
Input 1	1	1	1	1	1	1	1,00	0,88
Output 1	1	0	1	1	1	1	0,83	0,92
Processing 1	1	0	1	0	0	1	0,50	0,88
Output 2	1	0	1	1	1	1	0,83	
Output 3	1	1	1	1	1	1	1,00	
Processing 2	1	1	1	1	1	1	1,00	
Output 4	1	1	1	1	1	1	1,00	
Input 2	1	1	1	1	1	1	1,00	
Input 3	1	1	1	1	0	1	0,83	
Processing 3	1	1	1	1	1	1	1,00	
Input 4	0	1	0	1	1	1	0,67	
Processing 4	1	1	1	1	1	1	1,00	
Aufgabe 3								
Software 1	1	0	1	1	1	1	0,83	0,72
Software 2	0	0	1	1	1	1	0,67	
Software 3	0	1	1	1	1	0	0,67	
Hardware 1	1	0	1	1	1	1	0,83	0,61
Hardware 2	0	0	1	1	1	1	0,67	
Hardware 3	0	0	0	0	1	1	0,33	
Umgebung 1	1	1	1	1	1	1	1,00	0,67
Umgebung 2	1	0	0	1	1	1	0,67	
Umgebung 3	0	1	0	0	1	0	0,33	

Anhang E

Transkriptionsregeln

Die hier aufgeführten Transkriptionsregeln (vgl. Tabelle E.1 und E.2) orientieren sich an einem Leitfaden (Dresing und Pehl, 2012), der sich auf verbale Äußerungen konzentriert. Die Sprache wird dabei geglättet und auch nonverbale Aktionen werden berücksichtigt.

Tabelle E.1: Transkriptionsregeln für verbale Äußerungen

JA	Starke Betonung
...	Satz wird nicht beendet
- B	B unterbricht A in Satz und fährt mit eigenem Satz fort
(Wort ?)	Unsicherheit in der Transkription
(unv., Tür geht auf)	Unverständliche Stellen und ggf. Grund dafür

Tabelle E.2: Transkriptionsregeln für nonverbale Äußerungen

[]	Nonverbale Aktionen
()	Nicht gesagtes/ausgelassenes Wort, was wichtig ist, um den Sinn zu verstehen bzw. falls eine Veränderung der Wortreihenfolge zwingend notwendig ist, kann diese somit geändert werden
[...]	Teile des Transkripts wurden entfernt, da sie an dieser Stelle nicht relevant sind
[überlegen]	Unterhaltung pausiert zum Nachdenken
[programmieren]	Schreiben oder verändern das Programm
[testen Programm auf Roboter]	Drücken in der Programmierumgebung auf die grüne Flagge (Startbutton o. Ä.), um das Programm zu starten
[arbeiten am Roboter]	Interagieren mit dem Roboter
[erhalten eine Nachricht auf Tablet]	Signalton des Tablets ertönt; gucken auf das Tablet und sagen, dass sie eine Nachricht erhalten haben
[A und B tauschen Sitzplätze]	ProbandInnen wechseln die Sitzplätze; zumeist verbunden mit einem Wechsel der Tätigkeiten (Programmieren oder Konstruieren)

Ich erkläre, dass ich die Dissertation selbstständig und nur unter Verwendung der von mir gemäß § 7 Abs. 3 der Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät, veröffentlicht im Amtlichen Mitteilungsblatt der Humboldt-Universität zu Berlin Nr. 126/2014 am 18.11.2014 angegebenen Hilfsmittel angefertigt habe.

Berlin, den 31.07.2018 Sandra Schulz